APRENDE A PROGRAMAR

INICIACIÓN AL BASIC

VIAJE ESPACIAL



Ediciones Generales Anaya

PARA SPECTRUM Y COMMODORE

Diseño: Cooper West
Edición: James McCarter
Programas: Steve Rodgers
Marcus Milton
Ilustraciones: Lionel Jeans
Rob Shone

Traducción: Pedro Garre del Olmo

Agradecemos su colaboración a
J. David García Rodríguez

Título original: BEGINNING BASIC

Villafranca, 22. 28028 Madrid
I.S.B.N.: 84-7525-314-8
Depósito legal: M. 39.270/1985
Impreso por: Melsa. Ctra. de Fuenlabrada a Pinto, km. 21,800. Pinto (Madrid)
Impreso en España - Printed in Spain


APRENDE A PROGRAMAR

INICIACIÓN AL BASIC



Gary Marshall

Ediciones Generales Anaya



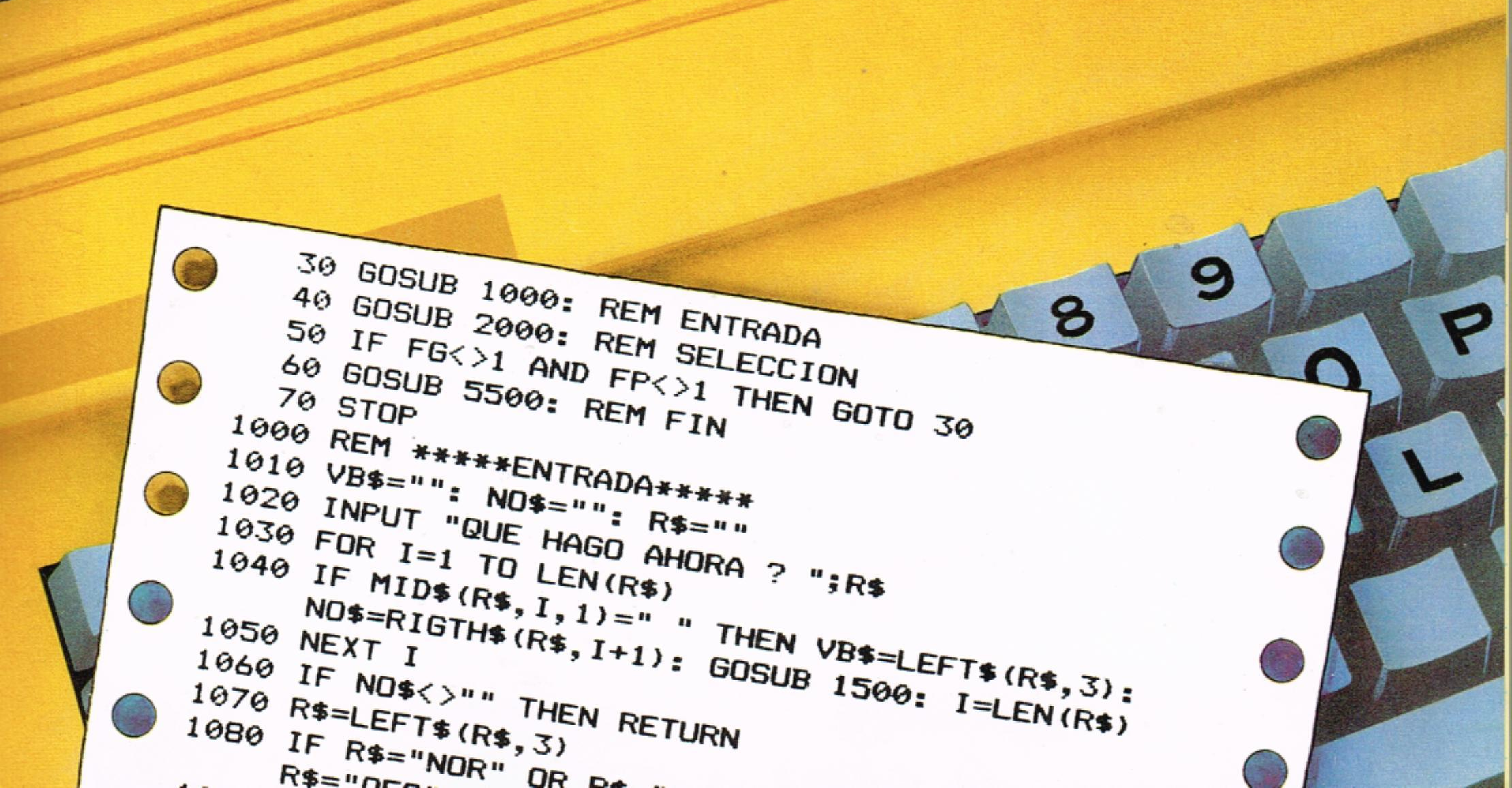
Introducción

Antes de que un ordenador pueda realizar una tarea, debe tener un programa, es decir, un conjunto de instrucciones que le explican exactamente lo que deseas que haga. Puedes comprar programas en cintas o discos, o teclear aquéllos que se publican en revistas o libros, pero al poco tiempo a la mayoría de la gente le gusta escribir sus propios programas. Una vez que sepas programar, podrás hacer que tu ordenador realice todo tipo de cosas para ti.

Este libro contiene tres programas de juegos, basados en el tema de un viaje espacial. Se incluyen versiones para Spectrum y para Commodore; debes leer la parte relativa a tu ordenador, pues a menudo la instrucción que realiza una determinada tarea varía ligeramente según el ordenador utilizado. Cada programa se desarrolla paso a paso, de forma que puedas ver exactamente cómo está hecho y cómo funciona. Al término de cada programa se sugieren unas modificaciones que puedes hacer por ti mismo.

Al final del libro tienes una sección que te muestra cómo se pueden juntar los tres programas para producir un juego con tres fases. Se da también el listado completo de este juego. Hay también un glosario con los términos informáticos usados en el libro, para que puedas saber qué significan.

Contenido



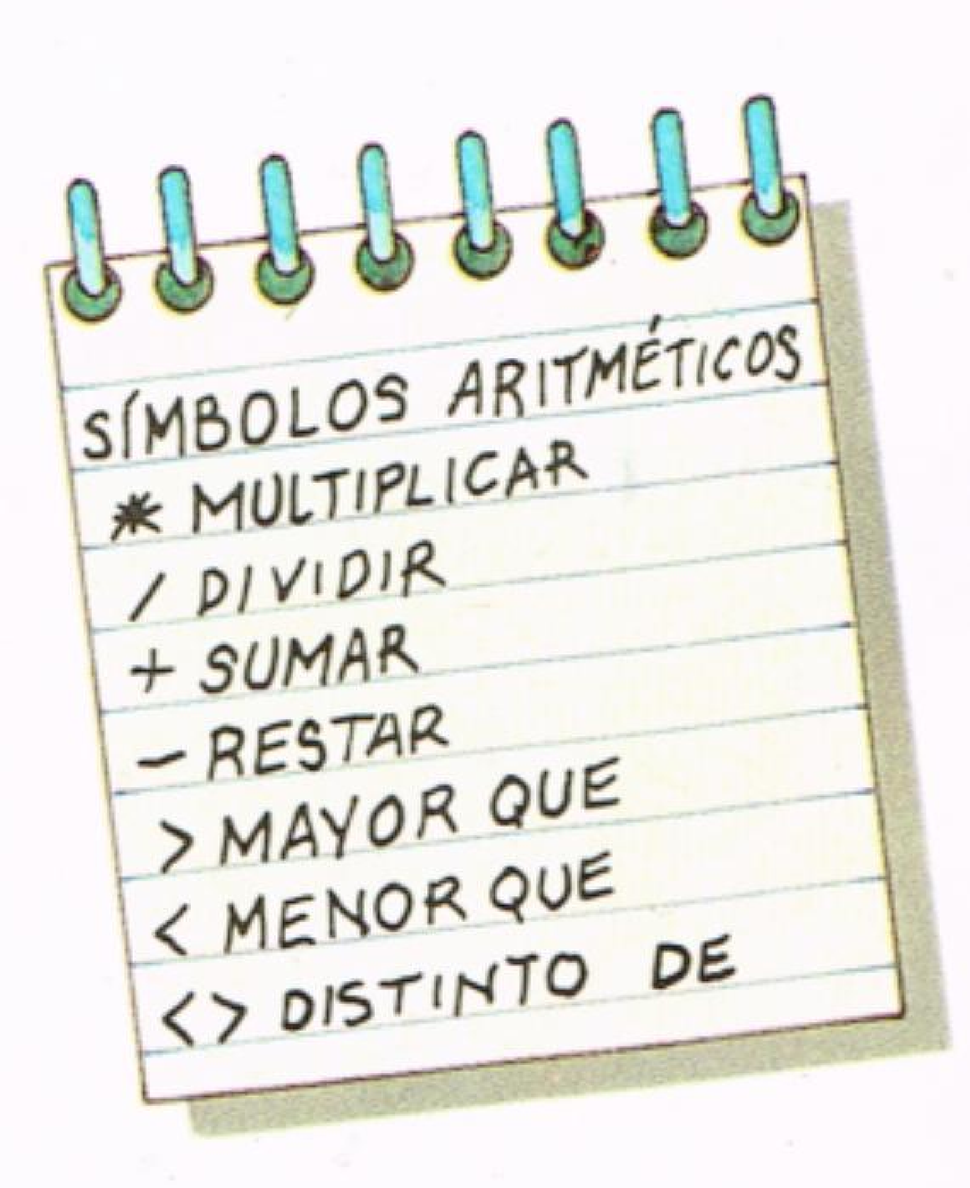
30 GOSUB 1000: REM ENTRADA
40 GOSUB 2000: REM SELECCION
50 IF FG<>1 AND FP<>1 THEN GOTO 30
60 GOSUB 5500: REM FIN
70 STOP
1000 REM *****ENTRADA*****
1010 VB$="": NO$="": R$=""
1020 INPUT "QUE HAGO AHORA ? ";R$
1030 FOR I=1 TO LEN(R$)
1040 IF MID$(R$,I,1)=" " THEN VB$=LEFT$(R$,3):
NO$=RIGTH$(R$,I+1): GOSUB 1500: I=LEN(R$)
1050 NEXT I
1060 IF NO$<>"" THEN RETURN
1070 R$=LEFT$(R$,3)
1080 IF R$="NOR" OR
8
9
O
P
L

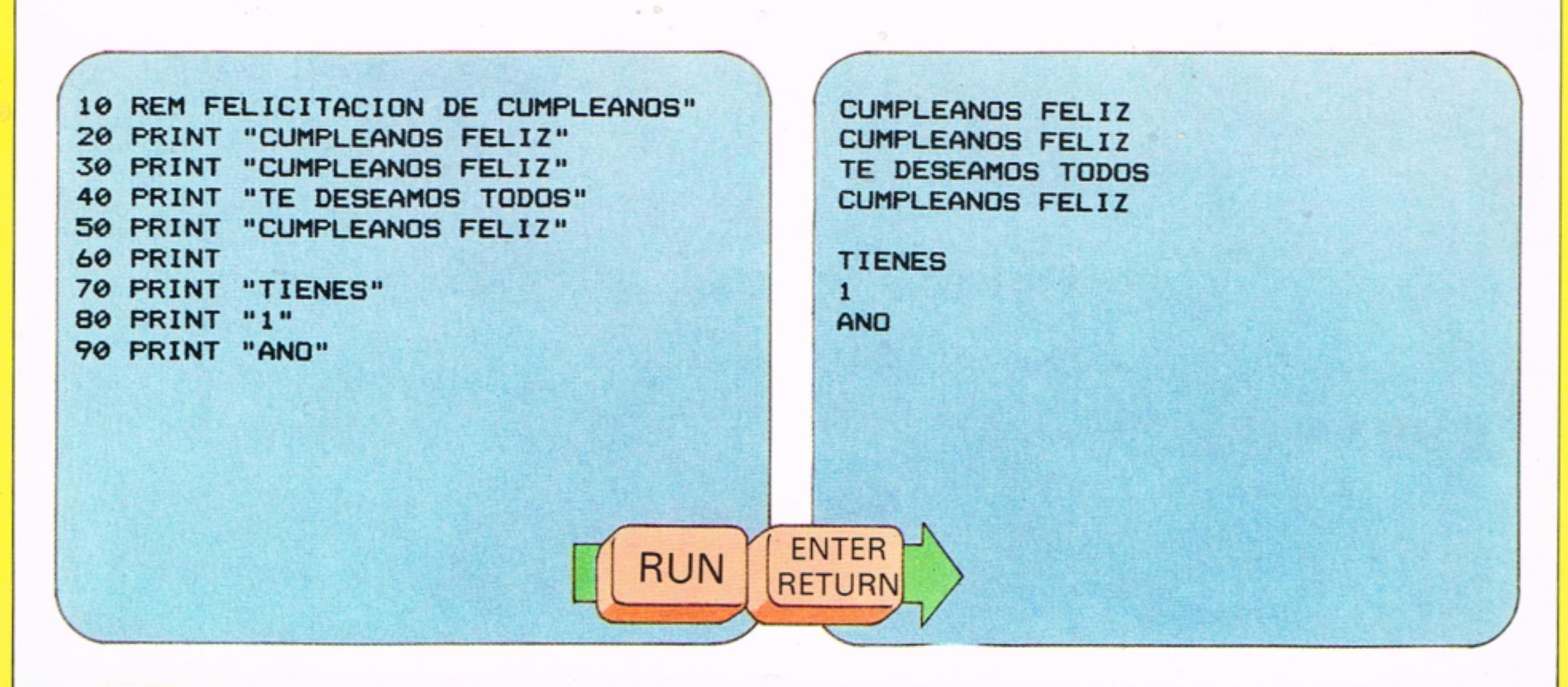
Introducción al BASIC

La mayoría de los programas para ordenadores personales están escritos en un lenguaje llamado BASIC. Como cualquier otro lenguaje, el BASIC tiene unas reglas que han de seguirse para que el programa funcione. Por ejemplo, la instrucción de BASIC **PRINT** escribe información en la pantalla, pero todo el texto debe estar metido entre comillas, como en la primera línea del programa indicado más abajo. Prueba las otras líneas y verás lo que ocurre. Notarás que las líneas dos y tres producen resultados muy diferentes; es así porque el ordenador trata de forma distinta las expresiones y las palabras. Los símbolos aritméticos del BASIC se muestran en la nota de la derecha.

```
PRINT "FELIZ CUMPLEANOS"
PRINT "15*2"
PRINT 15*2
```

En un programa, el ordenador realiza cada instrucción siguiendo un orden lógico; un programa no podría funcionar si el ordenador ejecutara cada instrucción tan pronto como ésta fuera tecleada. Al dar a cada instrucción un número de línea, le decimos al ordenador que la almacene en su memoria hasta que el programa sea ejecutado (**RUN**). Los números de línea usualmente van de diez en diez, de forma que las tres primeras líneas de un programa suelen ser 10, 20 y 30. El ordenador considera cada instrucción según su número de línea, y se pueden añadir instrucciones adicionales asignándoles números de línea como 15, 25, etc., según convenga. Introduce el corto programa indicado abajo. Observa que la línea 10 no escribe nada en la pantalla. Ésto se debe a que la orden **REM** se utiliza para comentarios, y el ordenador ignora todo lo situado tras ella.



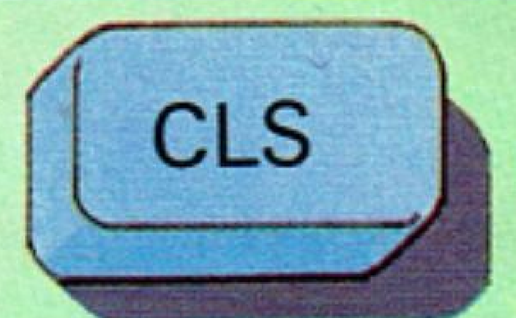


Algunas cosas sobre tu ordenador

Antes de empezar a trabajar en un programa, es una buena idea experimentar con tu ordenador para conocer el teclado. No importa lo que teclees, no puedes producir ningún daño al ordenador; la mayoría te responderán con un mensaje de error, como «Mistake» o «Syntax error». El Spectrum indica que hay un error en una línea de un programa colocando una interrogación intermitente delante del error, y la línea no puede ser introducida hasta que el error esté corregido. En concreto, aprende cómo se usan las teclas de edición, de forma que puedas borrar los errores que cometas. Los manuales que se acompañan con el ordenador pueden ayudarte en este sentido. No te preocupes si hay algunas teclas que no entiendes; aprenderás lo que hacen cuando las necesites. He aquí algunas de las teclas e instrucciones más importantes.



Esta tecla (**ENTER** para el Spectrum y **RETURN** para el Commodore) se usa para introducir una línea del programa en la memoria del ordenador. Uno de los errores más comunes es escribir una línea larga y olvidar pulsar esta tecla. La única solución es volver a escribir la línea.



Esta tecla del Spectrum, junto con **ENTER**, borra toda la pantalla.



Esta es una tecla del Spectrum. Esta instrucción borra toda la memoria.



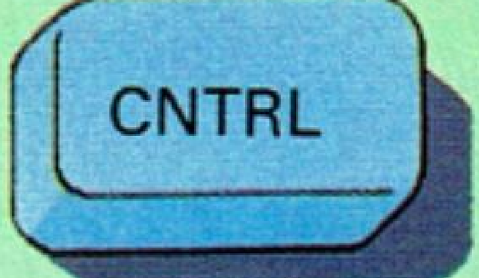
Si escribes **LIST** en el Spectrum y después pusas **ENTER**, el programa que has escrito aparecerá en la pantalla. Cuando **LIST** va seguida por un número, listará el programa a partir del número de línea que indica este valor.



Esta tecla del Commodore interrumpe un programa mientras se está ejecutando. Si el programa está esperando un **INPUT**, entonces debes pulsar esta tecla junto con **RESTORE** para interrumpir el programa.



Esta es la tecla Commodore y tiene diversas funciones. Da acceso a los segundos ocho colores disponibles y a los símbolos gráficos del lado izquierdo de cada tecla.



Esta tecla, abreviatura de **CONTROL**, realiza varias funciones en el Commodore. Se usa para obtener los primeros ocho colores disponibles.

Preparación de un programa

Antes de empezar a escribir un programa en BASIC debes saber exactamente qué quieres que haga el ordenador. El primer paso es escribir un esquema del programa en español normal. Por ejemplo, un esquema para el primer programa de este libro podría ser: «La idea del juego es adivinar un número aleatorio. Si se consigue, el ordenador simula el despegue de un cohete. Sólo hay 10 intentos. Las instrucciones de juego deben aparecer claramente en la pantalla».

También debes planear cómo quieres sacar las instrucciones por la pantalla, para que queden claras. Para esto es muy útil un papel cuadriculado, pues la pantalla del ordenador está también dividida en líneas horizontales y verticales, aunque el número exacto de éstas varía de una máquina a otra. El planteamiento inicial para el segundo juego de este libro, en el cual debes pilotar un cohete a través de un campo de asteroides, se muestra en la nota adjunta, así como un plan para la salida por pantalla.

Una vez hechos estos preparativos iniciales puedes pasar a la siguiente etapa: la construcción del diagrama de flujo. El diagrama de flujo te permitirá separar el esquema inicial del programa en cada uno de sus pasos lógicos.

Diagrama de flujo

Los diagramas de flujo separan los problemas en un conjunto de instrucciones y decisiones. El indicado a continuación es para el primer programa de este libro. Las cajas con forma de rombo indican que se toma una decisión; por ejemplo, si el número es demasiado alto o si ya has usado todos tus intentos. A las preguntas se debe responder con sí o no, y dependiendo de esta respuesta, se realiza el paso siguiente correspondiente. Las cajas cuadradas se usan para indicar que se hace una operación, como escribir las instrucciones de juego. Los diagramas de flujo te ayudan a hacer las preguntas adecuadas, de forma que puedas dar las instrucciones correctas al ordenador. Estudiando el diagrama de flujo, es posible ver si se ha olvidado alguna etapa.

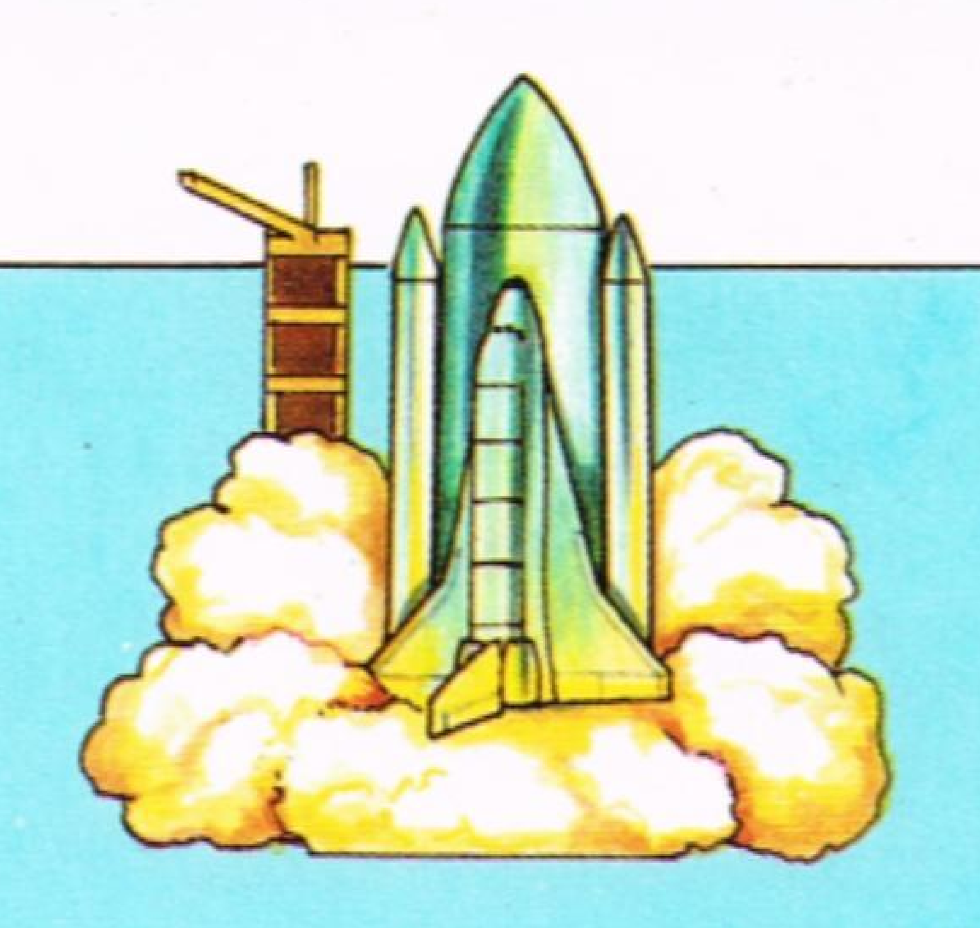
Una vez que el diagrama de flujo funciona, lo único que resta por hacer es convertir cada instrucción en órdenes del lenguaje BASIC.

Grabación de programas en cinta

Por muy bien que llegues a conocer un programa, se tarda mucho tiempo en teclearlo en el ordenador cada vez que se quiera ejecutar. Además, muy probablemente, cometerás errores al escribirlo, por lo que el programa deberá ser revisado y corregido cuidadosamente, o depurado, como se dice en jerga informática. Por tanto, una vez que tengas el programa funcionando querrás almacenarlo en una cinta para uso posterior. Es una tarea sencilla y las instrucciones para cargar y salvar programas se indican en la nota adjunta. Es una buena idea salvar dos versiones de tu programa antes de desconectar el ordenador, por si la primera no se graba correctamente, como ocurre a veces. Haz la primera grabación y coloca el contador de la cinta en veinte, por ejemplo, antes de efectuar la segunda. Escribe en la cinta el nombre del programa y su posición: «Copia10» para la primera versión, «Copia20» para la segunda y así sucesivamente. Por esto es importante colocar el contador de vueltas del cassette a cero antes de empezar. Con las cintas adecuadamente rotuladas, puedes encontrar rápidamente un programa cuando quieras usarlo de nuevo.

SALVAR PROGRAMAS
SPECTRUM: TECLEA SAVE "NOMBRE"
PULSA ENTER
PULSA REC/PLAY EN LA GRABADORA
PULSA CUALQUIER TECLA
COMMODORE: TECLEA SAVE "NOMBRE" -
PULSA RETURN - PULSA REC/PLAY EN LA
GRABADORA - LA PANTALLA SE BORRA -
CUANDO TERMINE, REBOBINA Y TECLEA
VERIFY "NOMBRE" PULSA RETURN,
PULSA PLAY - LA PANTALLA SE
BORRA - CUANDO SE TERMINE
APARECERÁ EN LA PANTALLA
O.K.

CARGA DE PROGRAMAS
SPECTRUM
TECLEA LOAD "NOMBRE"
PULSA PLAY EN LA GRABADORA
COMMODORE
TECLEA LOAD. PULSA RETURN
PULSA PLAY - LA PANTALLA
SE BORRA - LA PANTALLA
REAPARECE - SALE FOUND
NOMBRE. PULSA LA TECLA
COMMODORE, O ESPERA 30
SEGUNDOS - LA PANTALLA
SE BORRA - LA PANTALLA
REAPARECE - ¡CARGADO!



LANZAMIENTO DE UN COHETE

Eres el comandante de una nave espacial. Para hacer despegar la nave, debes indicar al ordenador central el nivel de energía necesario. El ordenador te dirá si tu número es demasiado alto o demasiado bajo, y si lo adivinas aparecerá el panel de control de la nave mostrándote cómo se incrementa la velocidad hasta que se alcanza la órbita.
¡Buena suerte!

```
BIENVENIDO A BORDO
CAPITAN GORDON

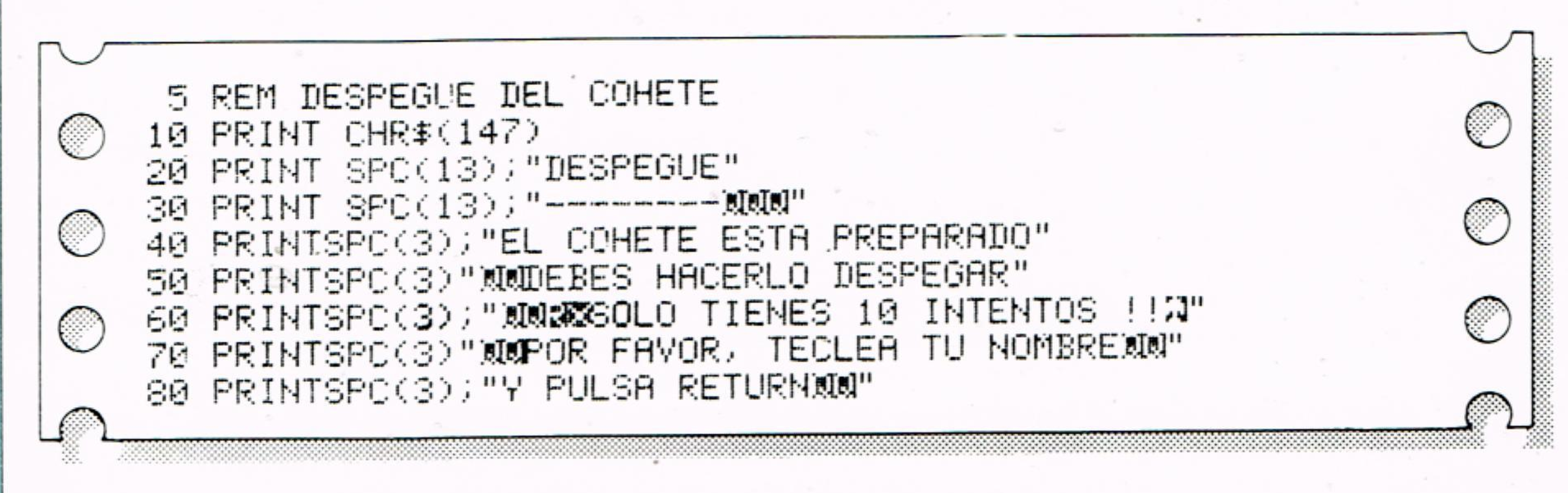
PARA HACER DESPEGAR EL COHETE
DEBES ADIVINAR EL NIVEL DE ENERGIA
NECESARIO (UN NUMERO ENTRE 1 Y 100)

TECLEA EL NUMERO Y PULSA ENTER
INTENTO NUMERO 2

50

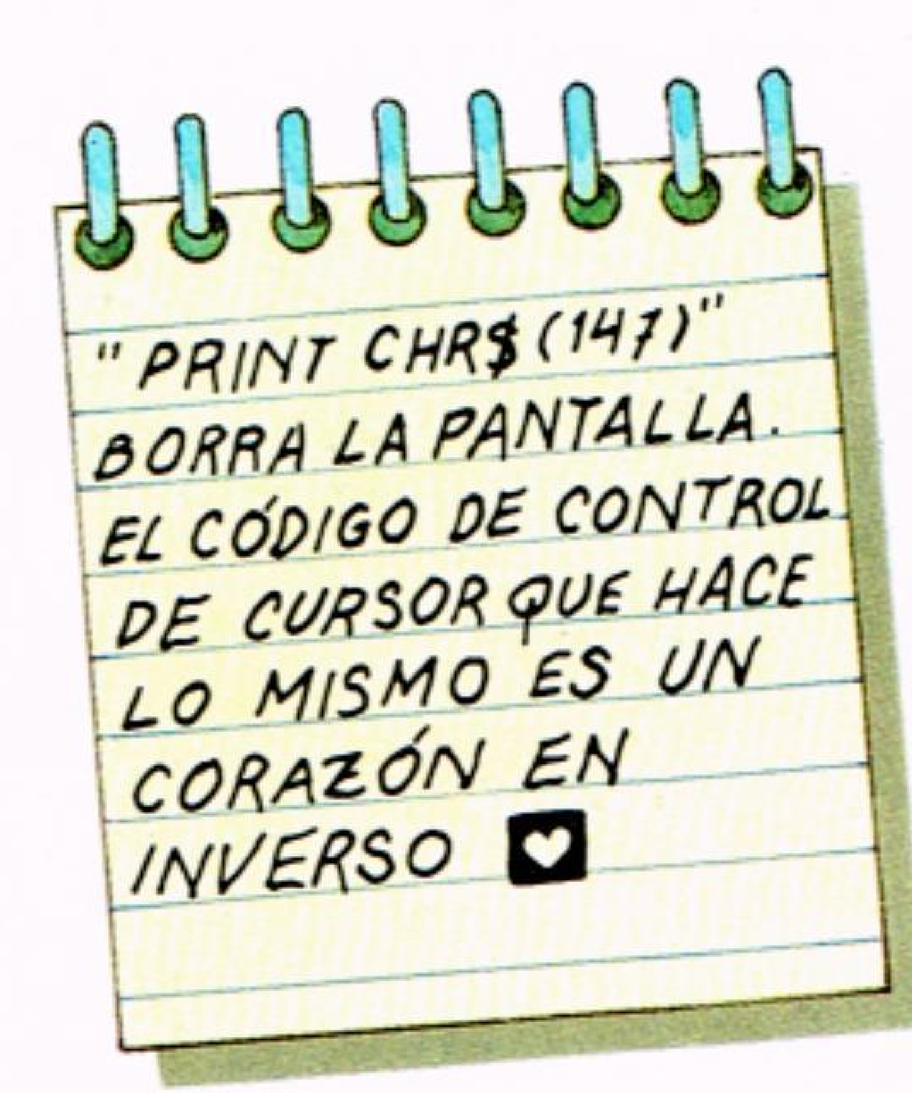
DEMASIADO ALTO  PRUEBA OTRA VEZ
```

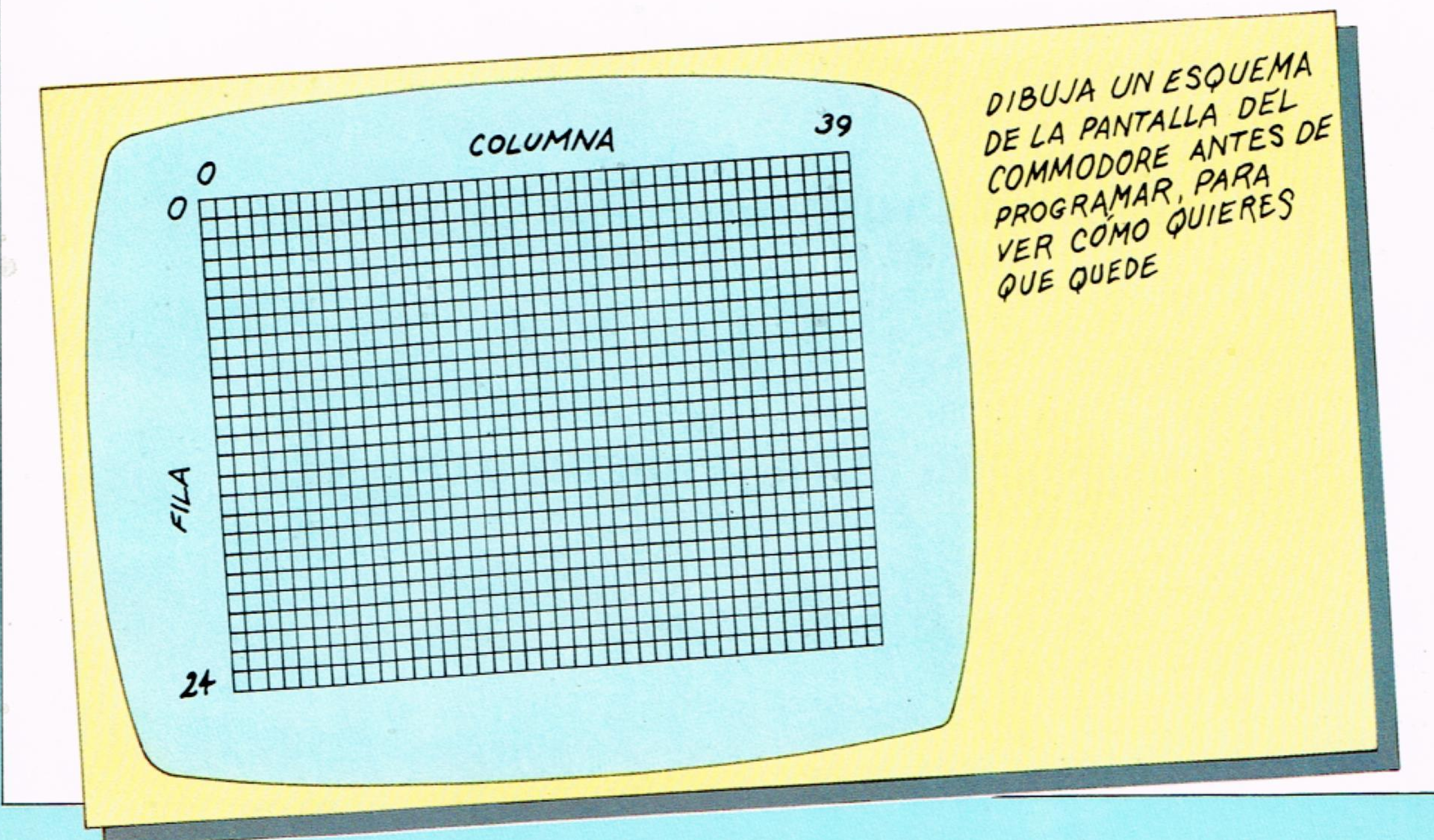
Este programa utiliza la capacidad del ordenador para generar números aleatorios. La mayoría de las líneas del programa son instrucciones para el jugador, que irán apareciendo en la pantalla en el momento apropiado del juego. Las primeras nueve líneas del programa crean la pantalla inicial.



```
 5 REM DESPEGUE DEL COHETE
10 PRINT CHR$(147)
20 PRINT SPC(13);"DESPEGUE"
30 PRINT SPC(13);"---------■■■"
40 PRINTSPC(3);"EL COHETE ESTA PREPARADO"
50 PRINTSPC(3)"■■DEBES HACERLO DESPEGAR"
60 PRINTSPC(3);"■■■■SOLO TIENES 10 INTENTOS !!■"
70 PRINTSPC(3)"■■POR FAVOR, TECLEA TU NOMBRE■■"
80 PRINTSPC(3);"Y PULSA RETURN■■"
```

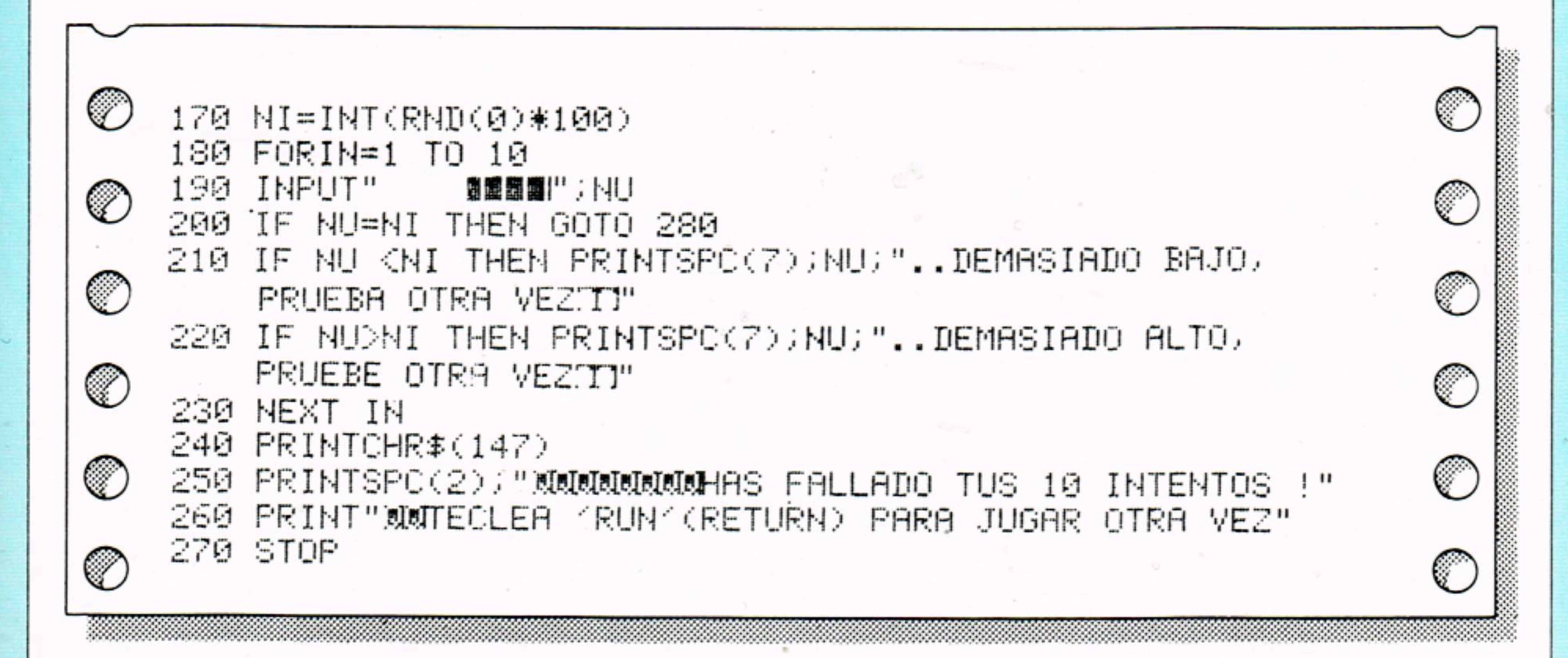
En la línea 10 se limpia la pantalla, usando **CHR$(147).** Para hacer que todo el texto aparezca en el centro de la pantalla, se usa la instrucción **SPC** (abreviatura de espacio). A continuación de **SPC** viene, entre paréntesis, el número de espacios que deben aparecer antes de la impresión del texto. Para colocar los mensajes a lo largo de la pantalla se utilizan códigos de control del cursor. Estos se obtienen tecleando primero unas comillas y luego pulsando la tecla de cursor arriba/abajo. Con esto hacemos que aparezca en la pantalla una «Q» en inverso, en lugar de mover el cursor. Por tanto, los tres símbolos de control del cursor de la línea 30 hacen que queden tres líneas en blanco entre el subrayado y el mensaje de la línea 40. En las líneas 50 a 80, en cambio, los símbolos aparecen antes del mensaje que va a ser escrito, para dejar dos líneas en blanco.





```
 90 INPUTNO$
100 PRINT CHR$(147)
110 PRINT"▮▮BIENVENIDO A BORDO CAPITAN ";NO$
120 PRINTSPC(5);"▮▮PARA HACER DESPEGAR EL COHETE"
130 PRINTSPC(5);"▮▮DEBES ADIVINAR EL NIVEL DE "
140 PRINTSPC(5);"▮▮ENERGIA NECESARIO"
150 PRINTSPC(5);"▮▮(UN NUMERO ENTRE 1 Y 100)"
160 PRINTSPC(5);"▮▮TECLEA EL NUMERO Y PULSA RETURN"
```

La línea 90 permite introducir el nombre del jugador cuando el programa se está ejecutando. La orden **INPUT** pone el cursor en modo intermitente, para avisar al usuario. El nombre tecleado se sitúa en una variable alfanumérica llamada **NO$**; el signo **$** indica al ordenador que acepte como entrada tanto letras como números. El ordenador imprime el nombre cuando se le pide con **PRINT NO$**, como en la línea 110. Observa que la cadena se encuentra fuera de las comillas, y que está precedida por un punto y coma. Después del **INPUT** se limpia la pantalla, y se imprimen las instrucciones contenidas en las líneas 110 a 160.

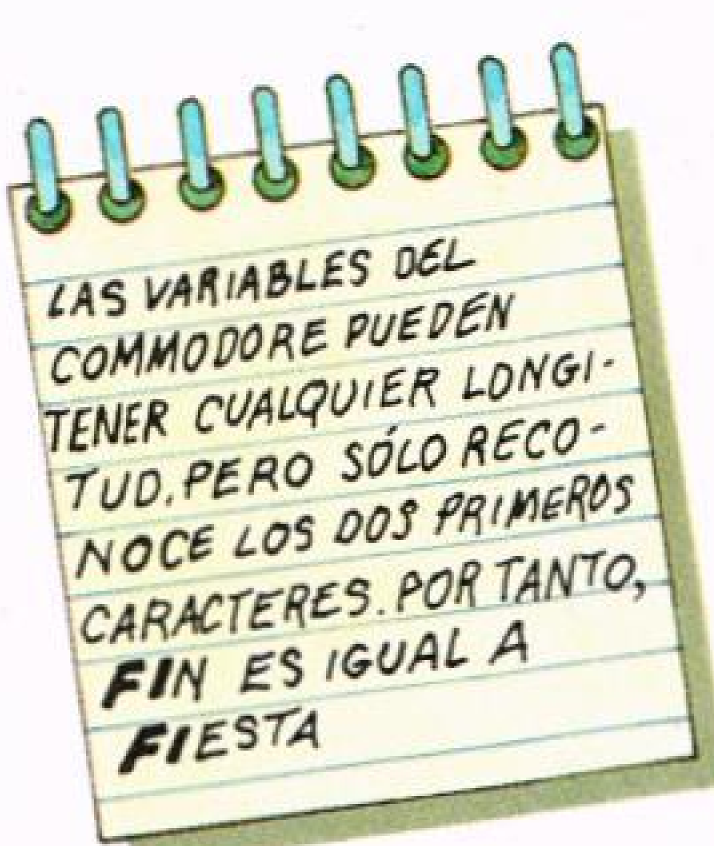


```
170 NI=INT(RND(0)*100)
180 FORIN=1 TO 10
190 INPUT"      ▮▮▮▮▮";NU
200 IF NU=NI THEN GOTO 280
210 IF NU <NI THEN PRINTSPC(7);NU;"..DEMASIADO BAJO,
    PRUEBA OTRA VEZ▮▮"
220 IF NU>NI THEN PRINTSPC(7);NU;"..DEMASIADO ALTO,
    PRUEBE OTRA VEZ▮▮"
230 NEXT IN
240 PRINTCHR$(147)
250 PRINTSPC(2);"▮▮▮▮▮▮▮▮HAS FALLADO TUS 10 INTENTOS !"
260 PRINT"▮▮TECLEA 'RUN'(RETURN) PARA JUGAR OTRA VEZ"
270 STOP
```

La instrucción **RND(0)** de la línea 170 genera un número aleatorio entre 0 y 1; éste se multiplica por 100 y se transforma en un número entero por medio de la función **INT.** Por último, dado que **INT** redondea por defecto los números decimales al entero más cercano, se le suma uno para asegurarse de que nunca será cero. Este número se almacena en la variable **NI** (abreviatura de nivel). La línea 180 limita el número de intentos a diez. Si el número es correcto, el programa saca el mensaje correspondiente de la línea 280; en caso contrario, las líneas 210 y 220 dicen al jugador si el número elegido era demasiado alto o demasiado bajo. Si se fallan los diez intentos, el programa va a las líneas 240 y 260 y se imprime el mensaje correspondiente.

```
280 PRINTCHR$(147)
290 AL=0
300 OB=10000
310 PRINTSPC(14);"▒▒▒▒ELEVACION!!!!"
320 PRINTSPC(16);"▒▒▒▒ALTITUD▒▒▒"
330 PRINTSPC(16);"▒"AL
340 FOR RE=1 TO 500
350 NEXT RE
360 AL=AL*2+1
370 IF AL<OB THEN GOTO 330
```

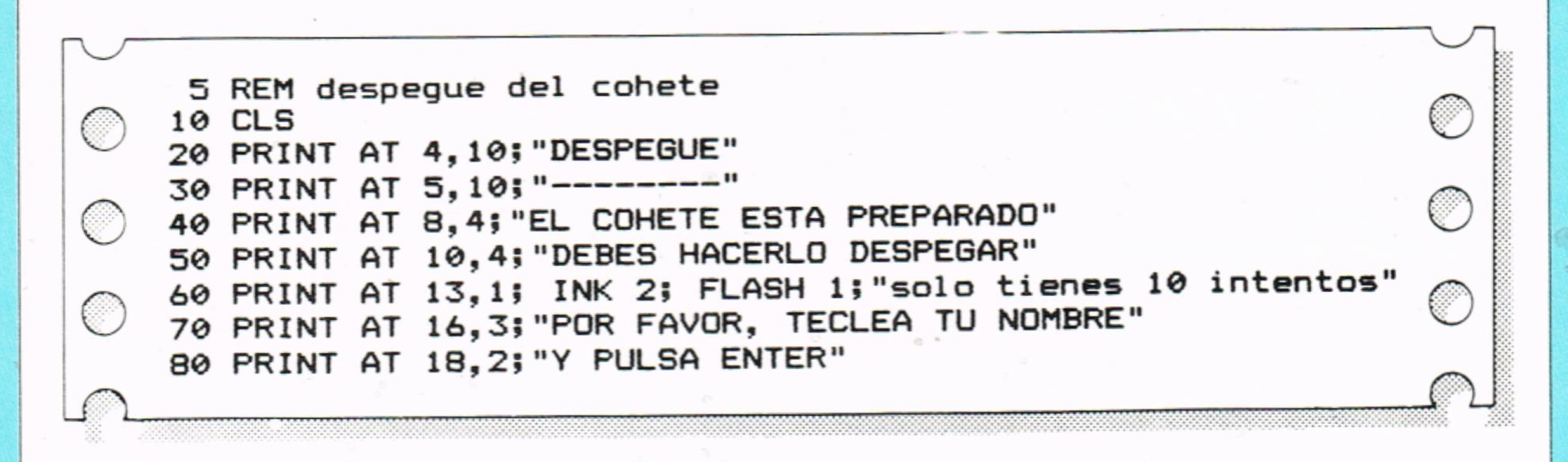
Al producirse el despegue aparece el panel de control y la altitud va aumentando a medida que la nave acelera para alcanzar su órbita. La línea 290 pone la altitud inicial a cero, y la 300 pone el objetivo de la órbita en 10000. Las siguientes tres líneas imprimen las palabras DESPEGUE y ALTITUD, así como el valor de la altitud, en las posiciones adecuadas de la pantalla para crear un panel de control. La línea 360 halla la nueva altitud, duplicando el valor anterior y sumándole uno. Dado que el ordenador trabaja muy deprisa, los valores de la altitud pasarían por la pantalla demasiado rápido para verlos claramente, si se permitiera al ordenador simplemente hacer el cálculo e imprimirlo. Para frenar al ordenador, las líneas 340 y 350 hacen un bucle de retardo. Estas órdenes obligan al ordenador a contar hasta 500 antes de hacer cualquier otra cosa; para producir una elevación más rápida disminuye el valor del retardo. La línea 370 dice al ordenador que vuelva a la 330, imprima el valor de altitud obtenido y vuelva a calcular un nuevo valor de altitud; esto se repetirá hasta que la altitud sea mayor que el objetivo para alcanzar la órbita.




```
380 PRINTCHR$(147)
390 FORFI=1TO15
400 PRINTSPC(11);"COHETE EN ORBITA!!!!"
410 NEXTFI
420 PRINT"▒▒  ENHORABUENA CAPITAN ";NO$
```

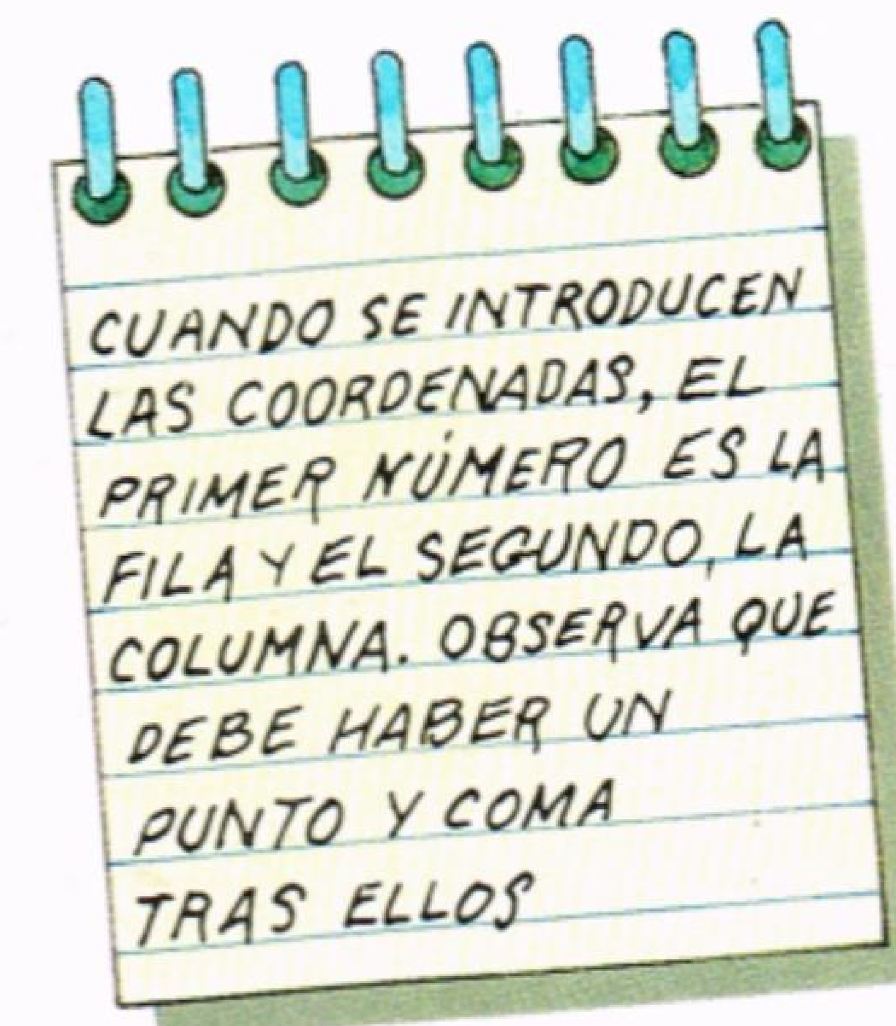
La sección final del programa imprime un mensaje de felicitación. En las líneas 390 a 410 se usa un bucle **FOR...NEXT** para escribir COHETE EN ORBITA! quince veces. Para ello utiliza la variable **FI** (abreviatura de fila), en este caso tomando valores desde 1 a 15, con lo que le dice al ordenador que escriba desde la fila 1 a la 15. Sin este bucle habría sido necesario copiar la línea 400 quince veces, cada una con un número de línea de programa distinto.

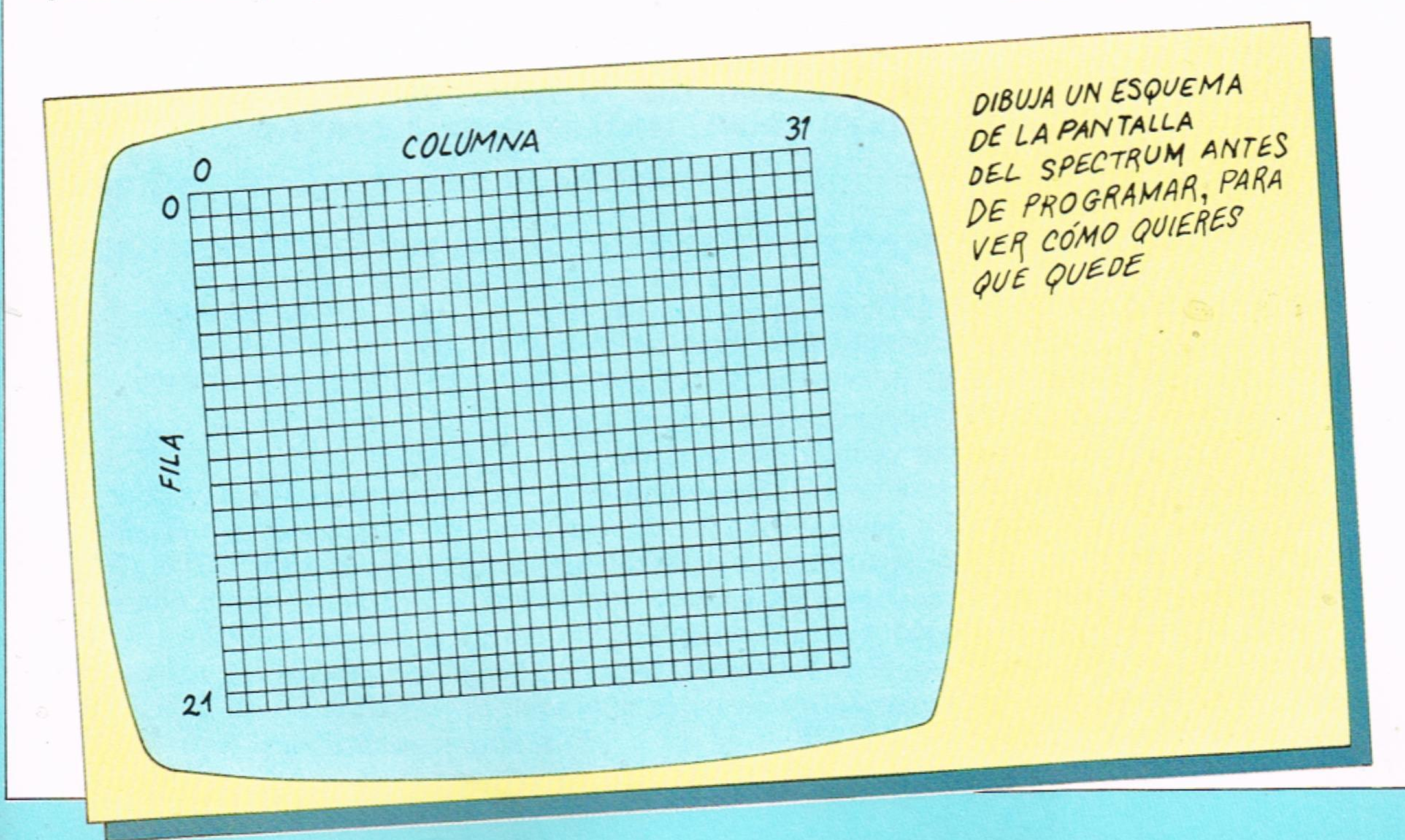
Este programa utiliza la capacidad del ordenador para generar números aleatorios. La mayoría de las líneas del programa son instrucciones para el jugador, que irán apareciendo en la pantalla en el momento apropiado del juego. Las primeras nueve líneas del programa crean la pantalla inicial.



```
 5 REM despegue del cohete
10 CLS
20 PRINT AT 4,10;"DESPEGUE"
30 PRINT AT 5,10;"--------"
40 PRINT AT 8,4;"EL COHETE ESTA PREPARADO"
50 PRINT AT 10,4;"DEBES HACERLO DESPEGAR"
60 PRINT AT 13,1; INK 2; FLASH 1;"solo tienes 10 intentos"
70 PRINT AT 16,3;"POR FAVOR, TECLEA TU NOMBRE"
80 PRINT AT 18,2;"Y PULSA ENTER"
```

La línea 5 usa un **REM** para dar un título al programa y la orden **CLS** de la línea 10 limpia la pantalla. Las siguientes líneas tienen la información que va a aparecer en la pantalla; la orden **PRINT** se puede usar para hacer esto, pero no produce una pantalla interesante (prueba y lo verás). Por eso se usa **PRINT AT** para posicionar los mensajes en la pantalla; los números que le acompañan indican el primero la fila, y el segundo la columna de la pantalla. El Spectrum tiene la pantalla dividida en 22 filas y 32 columnas. Observa que los números de fila y columna están separados por una coma y seguidos de un punto y coma. Con cualquier otro signo de puntuación, el programa no funcionaría correctamente.





```
 90 INPUT n$
100 CLS
110 PRINT "BIENVENIDO A BORDO": PRINT "CAPITAN ";n$
120 PRINT AT 4,1;"PARA HACER DESPEGAR EL COHETE"
130 PRINT AT 6,1;"DEBES ADIVINAR EL NIVEL DE"
140 PRINT AT 8,1;"ENERGIA NECESARIO"
150 PRINT AT 10,1;"(UN NUMERO ENTRE 1 Y 100)"
160 PRINT AT 12,1;"TECLEA EL NUMERO Y PULSA ENTER"
```

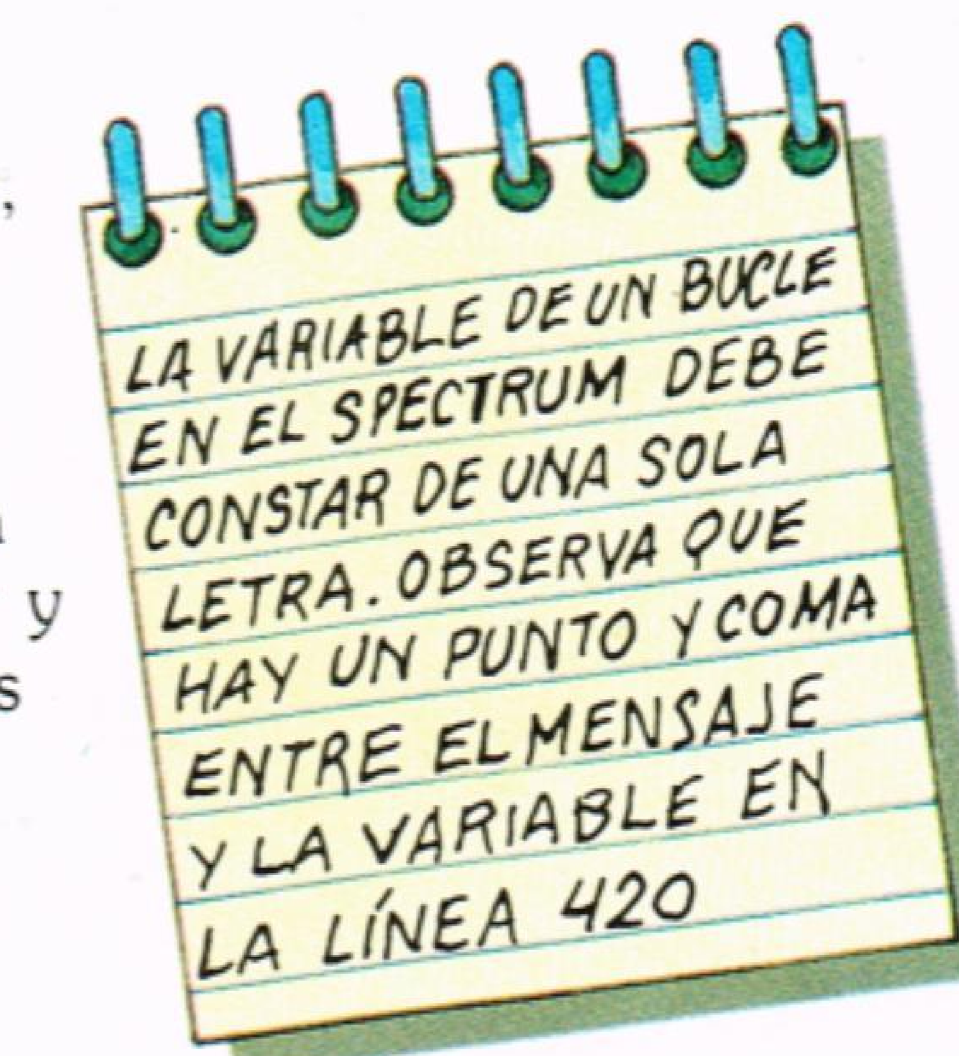
La línea 90 permite introducir el nombre del jugador cuando el programa se está ejecutando. La orden **INPUT** coloca el cursor en modo intermitente, para avisar al usuario. El nombre tecleado se sitúa en una variable alfanumérica llamada **n$**; el signo **$** indica al ordenador que acepte como entrada tanto letras como números; sin el, el ordenador sólo aceptaría números. Cuando al ordenador se le dice **PRINT n$**, como en la línea 110, escribe cualquier cosa guardada en **n$**. Observa que en esta línea, la cadena se encuentra fuera de las comillas y está precedida por un punto y coma. Después del **INPUT**, se limpia la pantalla y se deja preparada para el segundo conjunto de instrucciones de la líneas 110 a 160.

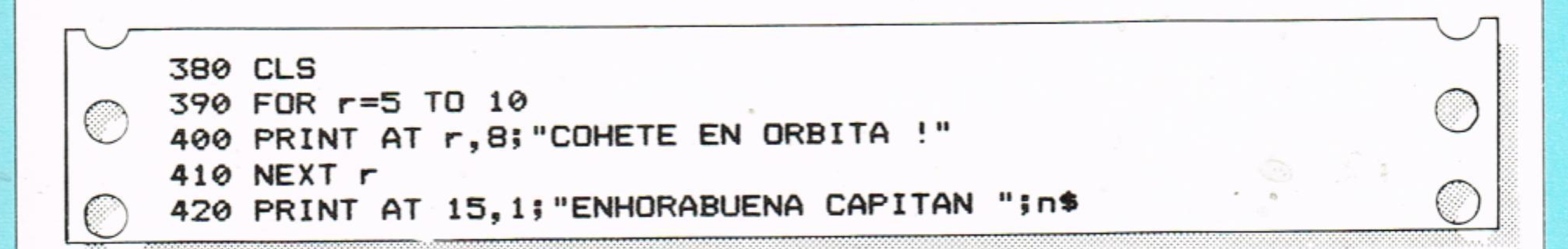
```
170 LET nivel=INT (RND*100)+1
180 FOR i=1 TO 10
190 INPUT num
195 PRINT AT 17,16;num;"    "
200 IF num=nivel THEN  GO TO 280
210 IF num<nivel THEN  PRINT AT 20,1;"DEMASIADO BAJO,
    PRUEBA OTRA VEZ"
220 IF num>nivel THEN  PRINT AT 20,1;"DEMASIADO ALTO,
    PRUEBA OTRA VEZ"
230 NEXT i
240 CLS
250 PRINT AT 4,0;"HAS FALLADO TUS 10 INTENTOS"
260 PRINT AT 6,0;"TECLEA 'RUN' (ENTER) PARA JUGAR";AT
    7,0;"OTRA VEZ"
270 STOP
```

La instrucción **RND** de la línea 170 genera un número aleatorio entre 0 y 1; éste se multiplica por 100 y se transforma en un número entero por medio de la función **INT**. Por último, dado que **INT** redondea por defecto los números decimales al entero más cercano, se le suma un 1 para asegurarse de que nunca será cero. Este número se almacena en la variable **nivel**. La línea 180 limita el número de intentos a diez. Si el número es correcto, el programa saca el mensaje correspondiente de la línea 280; en caso contrario, las líneas 210 y 220 dicen al jugador si el número elegido era demasiado alto o demasiado bajo. Si se fallan los diez intentos, el programa va a las líneas 240 y 260 y se imprime el mensaje correspondiente.

```
280 CLS
290 LET altitud=0
300 LET objetivo=10000
310 PRINT AT 2,10;"ELEVACION !!!!!"
320 PRINT AT 10,10;"ALTITUD"
330 PRINT AT 10,20;altitud
340 FOR r=1 TO 100
350 NEXT r
360 LET altitud=altitud*2+1
370 IF altitud<objetivo THEN  GO TO 330
```

Al producirse el despegue aparece el panel de control y la altitud va aumentando a medida que la nave acelera para alcanzar su órbita. La línea 290 pone la altitud inicial a cero, y la 300 pone el objetivo de la órbita en 10000. Las tres líneas siguientes imprimen las palabras ELEVACION y ALTITUD, así como el valor de la altitud en las posiciones adecuadas de la pantalla, para crear un panel de control. La línea 360 halla la nueva altitud, duplicando el valor anterior y sumándole uno. Como el ordenador trabaja muy deprisa, los valores de la altitud pasarían por la pantalla demasiado rápido para verlos claramente, si se permitiera al ordenador simplemente hacer el cálculo e imprimirlo. Para frenar al ordenaor. las líneas 340 y 350 hacen un bucle de retardo. Estas órdenes obligan al ordenador a contar hasta 100 antes de hacer cualquier otra cosa; para producir una elevación más rápida, disminuye el valor del retardo. La línea 370 dice al ordenador que vuelva a la 330, imprima el valor de altitud obtenido y vuelva a calcular un nuevo valor de altitud; esto se realizará hasta que la altitud sea mayor que el objetivo (en este caso 10000), con lo que se habrá alcanzado la órbita.

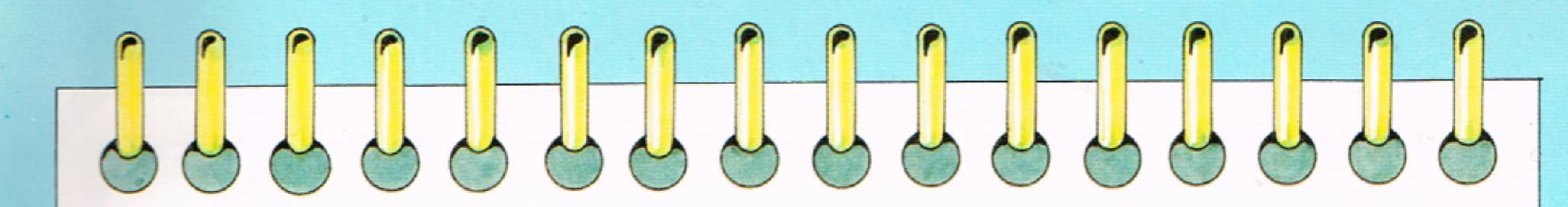




```
380 CLS
390 FOR r=5 TO 10
400 PRINT AT r,8;"COHETE EN ORBITA !"
410 NEXT r
420 PRINT AT 15,1;"ENHORABUENA CAPITAN ";n$
```

La sección final del programa imprime un mensaje de felicitación. En las líneas 390 a 410 se usa un bucle **FOR...NEXT** para escribir COHETE EN ORBITA! cinco veces. Para ello utiliza la variable **f** (abreviatura de fila), en este caso tomando valores desde 5 a 10, con lo que le dice al ordenador que escriba desde la fila 5 a la 10. Sin este bucle, habría sido necesario copiar la línea 400 cinco veces, una por cada fila que quisiéramos imprimir.





Mejora el programa

Una de las mejores formas de aprender programación es experimentar con los programas que escribes. Prueba a quitar el retardo de las líneas 340 y 350 y mira lo que ocurre. Cambia el número de intentos de que dispone el jugador, modificando las líneas 180 y 250. Apunta siempre las modificaciones que hagas, para que puedas devolver el programa a su forma primitiva.



Adición de sonido

COMMODORE

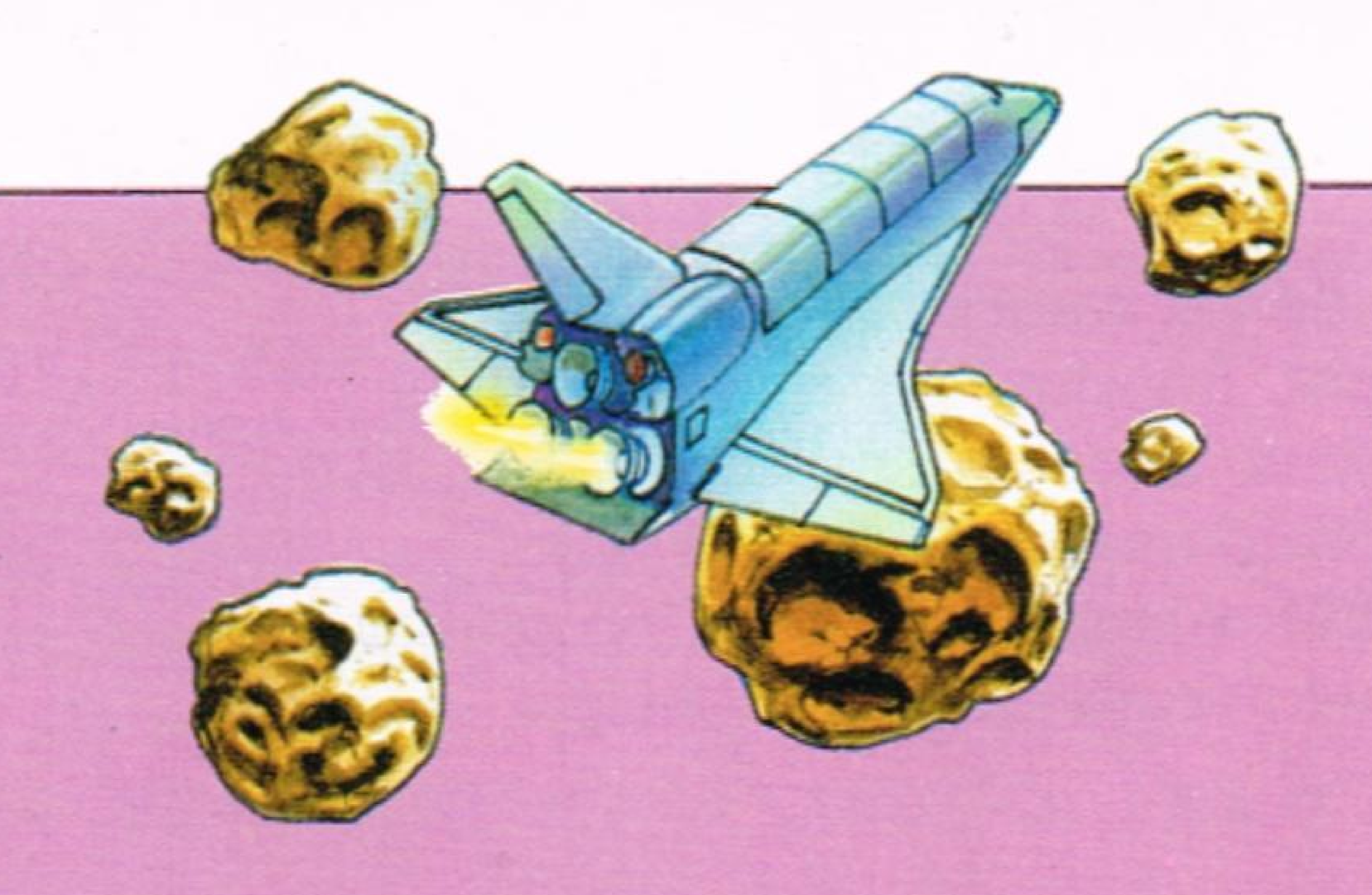
El Commodore puede producir sonidos extremadamente interesantes, pero las órdenes necesarias son muy complejas. El sonido se obtiene usando la instrucción **POKE**, seguida de dos números. La instrucción **POKE** está fuera del alcance de este libro, pero añade estas líneas a tu programa para producir sonidos en el despegue.

```
365 POKE54277,15:POKE54278,81:POKE54296,15:POKE54276,129
366 POKE54273,AL/65:FORT=0TO60:NEXT
375 POKE54276,128
```

SPECTRUM

El Spectrum utiliza la orden **BEEP** seguida de dos números; el primero indica la duración del sonido y el segundo el tono. Inserta una línea nueva, la 325 en este caso, y sustituye el bucle de retardo de las líneas 340 y 350 por las órdenes BEEP indicadas, para producir varios sonidos.

```
325 LET s=0
340 BEEP .1,s: BEEP .1,s: BEEP .1,s: BEEP .1,s
350 LET s=s+2
```

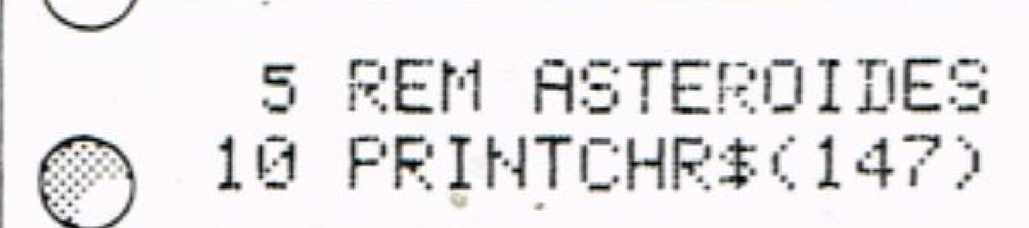


ASTEROIDES

El cohete está ya en órbita. De repente, te metes en una terrible tormenta de asteroides. Debes dar al ordenador central de la nave las instrucciones de vuelo adecuadas para atravesar la tormenta. Es imposible dar marcha atrás, y una colisión significará la total destrucción de tu nave.

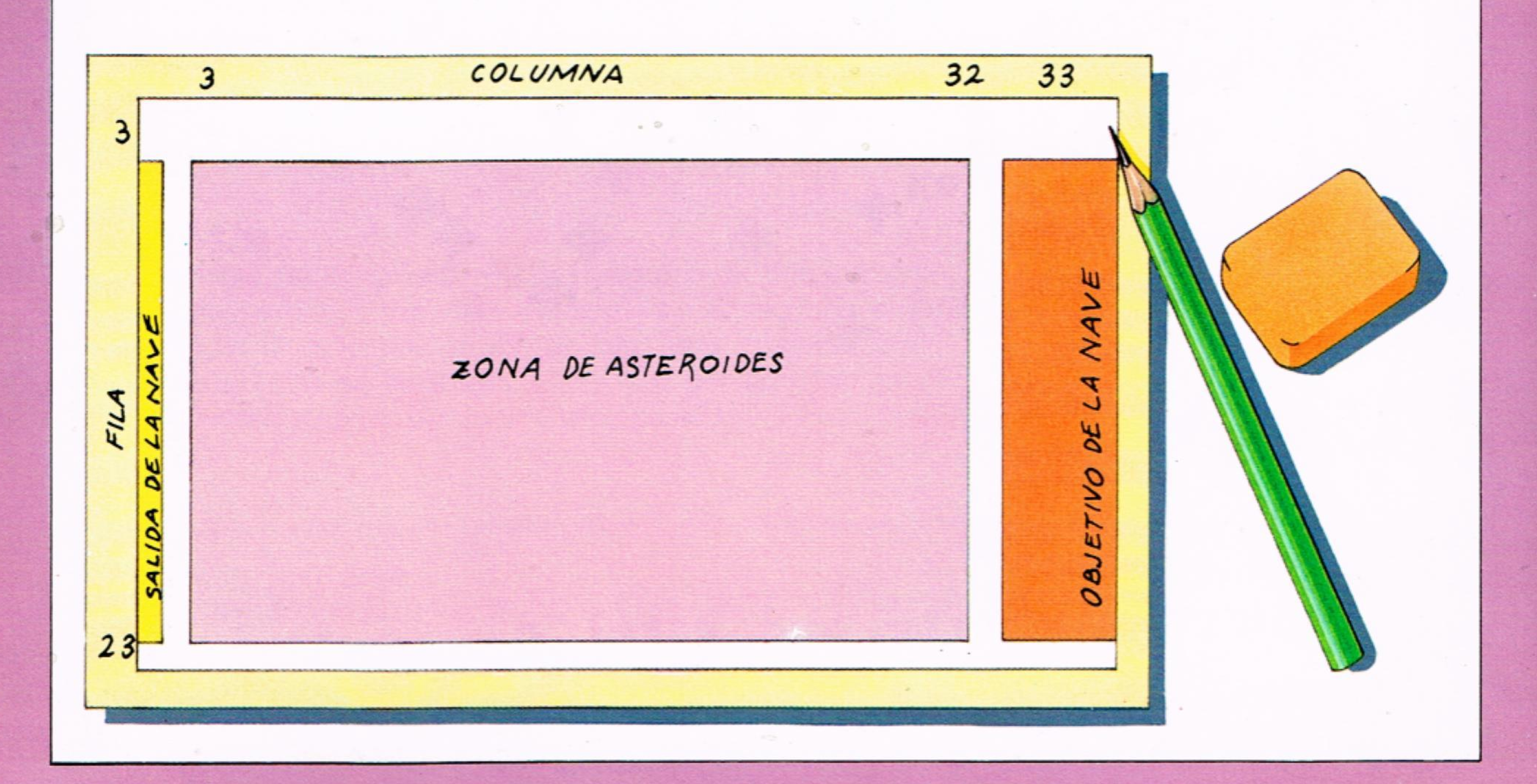
DIRECCION (norte, sur, este) "n"

Este programa usa algunos de los caracteres gráficos disponibles en el Commodore. Imprime los asteroides y el cohete en la pantalla, y permite al cohete moverse de acuerdo con las órdenes que se le den durante el juego. Como anteriormente, una línea **REM** asigna un título al programa y se limpia la pantalla de cualquier mensaje que hubiera usando **PRINT CHR$(147).**



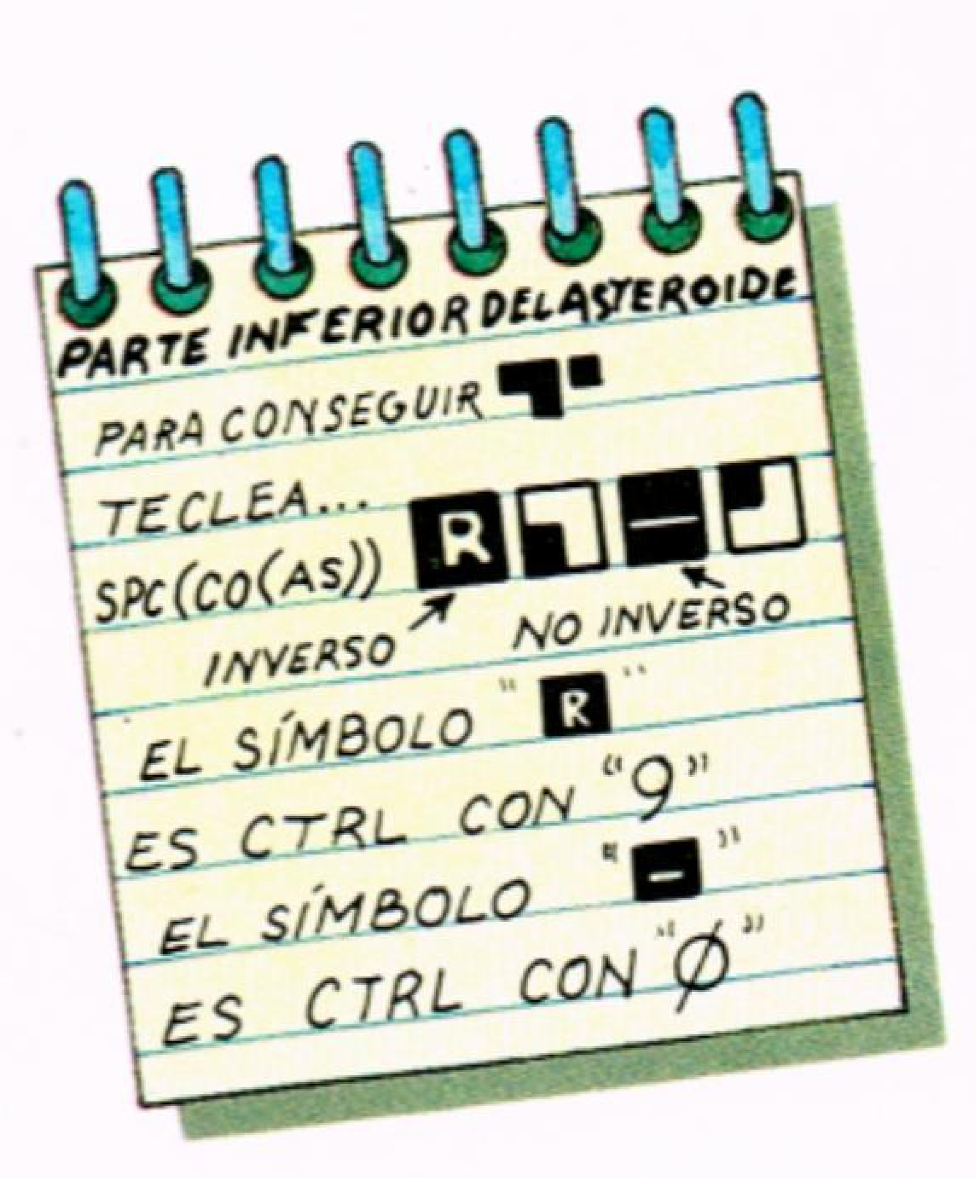
```
 5 REM ASTEROIDES
10 PRINTCHR$(147)
```

Lo primero, antes de escribir la sección del programa que dibujará los asteroides en posiciones aleatorias de la pantalla, es planear cómo va a quedar ésta. La pantalla del Commodore 64 tiene una anchura de 40 columnas y una altura de 25 filas, numeradas desde 0 a 39 y desde 0 a 24 respectivamente. La nave debe atravesar la pantalla de izquierda a derecha; por tanto, su punto de salida debe estar en la columna 0. El campo de asteroides ocupará la mayor parte de la pantalla (en este caso desde la columna 3 hasta la 32). La nave conseguirá su objetivo cuando pase la columna 33 sin haber chocado con ningún asteroide. Las tres primeras filas de la pantalla se reservan para imprimir instrucciones y mensajes indicando el desarrollo del juego, y para permitir la introducción de órdenes durante éste. Además, la nave no podrá volar por los bordes de arriba y abajo de la pantalla para evitar los asteroides. Teniendo en cuenta estas consideraciones se hace el planteamiento indicado más abajo. Puedes hacer otra planificación diferente, pero deberás cambiar algunos de los valores de las instrucciones **PRINT.**



```
20 DIM CO(50),FI(50)
25 UD$="SQQQQQQQQQQQQQQQQQQQQQQQQQQQQQ"
30 FORAS=1 TO 50
40 CO(AS)=INT(RND(0)*30)+3
50 FI(AS)=INT(RND(0)*20)+4
60 POKE 646,XX:PRINTLEFT$(UD$,FI(AS))SPC(CO(AS))"▖"
65 PRINTLEFT$(UD$,FI(AS)+1)SPC(CO(AS))"▚▘"
70 NEXT AS
```

En la línea 20, la orden **DIM** crea dos *matrices*, donde se van a guardar las coordenadas "columna" y "fila" de los asteroides. El número entre paréntesis indica el número de asteroides, y por tanto el tamaño necesario para la matriz. En la línea 25 se almacena en **UD$** una cadena de códigos de control del cursor, que se usará más tarde para poner el cursor en la fila deseada de la pantalla, como en la línea 110. Las líneas 30 a 70 son un bucle **FOR...NEXT** que hace que las órdenes de las líneas 40 a 65 se realicen para cada uno de los 50 asteroides, por turno. La variable de bucle **AS** indica el asteroide que toca. En las líneas 40 a 50 se generan una columna y una fila aleatorias que indican la posición de cada asteroide en la pantalla. La línea 60 hace la parte de arriba del asteroide, usando el símbolo gráfico de la tecla "D" Para obtener el símbolo, pulsa la tecla COMMODORE y la "D" a la vez. La línea 65 hace la parte de abajo, por medio de dos caracteres. El primero está en inverso (el carácter que aparece en el listado es el opuesto del que aparecerá en la pantalla). Mira la nota de al lado para aprender a obtener los caracteres inversos. El segundo carácter se consigue pulsando la tecla COMMODORE y la "V" al mismo tiempo. Prueba esta sección como se indica a continuación.





TRAS LA LÍNEA 70 TECLEA RUN Y APARECERÁ EL CAMPO DE ASTEROIDES. SI HAY ALGÚN ERROR, REVISA EL LISTADO DEL PROGRAMA

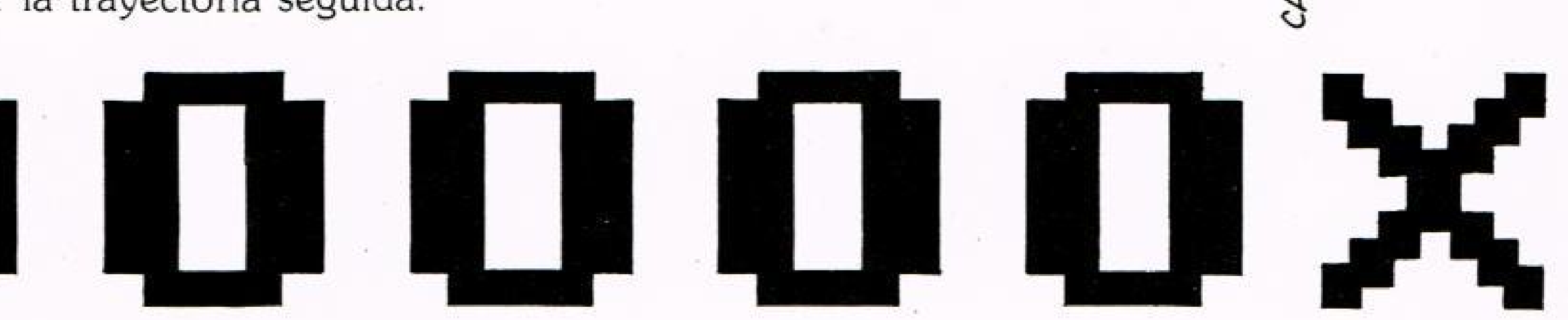
```
 80 TI$="000000"
 90 C$="                                      "
100 CO=0:FI=INT(RND(0)*20)+4
110 PRINTLEFT$(UD$,FI)SPC(CO)"X"
120 PRINT"■■"C$
130 PRINTC$
140 PRINT"■TIEMPO:"MID$(TI$,3,2)" MINUTOS   ";
145 PRINTRIGHT$(TI$,2)" SEGUNDOS"
150 IF VAL(MID$(TI$,4,1))>2 THEN 1350
```

En la línea 80 se pone a 0 la variable TI$, que representa el tiempo. Observa que tiene seis caracteres: los dos primeros cuentan las horas, los dos siguientes los minutos y los dos últimos los segundos. En la línea 140, la orden **MID$** saca los caracteres tercero y cuarto para imprimir los minutos y en la 145, **RIGHT$** extrae los dos últimos para imprimir los segundos. La línea 150 lleva al programa a la línea 350, donde se avisa que el tiempo se ha acabado, si el valor del cuarto carácter es mayor que dos, es decir, cuando han pasado dos minutos. La cadena de 38 espacios de la línea 90 se usa en las líneas 120 y 130 para borrar los valores antiguos impresos en la parte de arriba de la pantalla. Las líneas 100 y 110 imprimen un "X" en una posición aleatoria de la columna 0, para representar la nave dispuesta a iniciar su viaje.

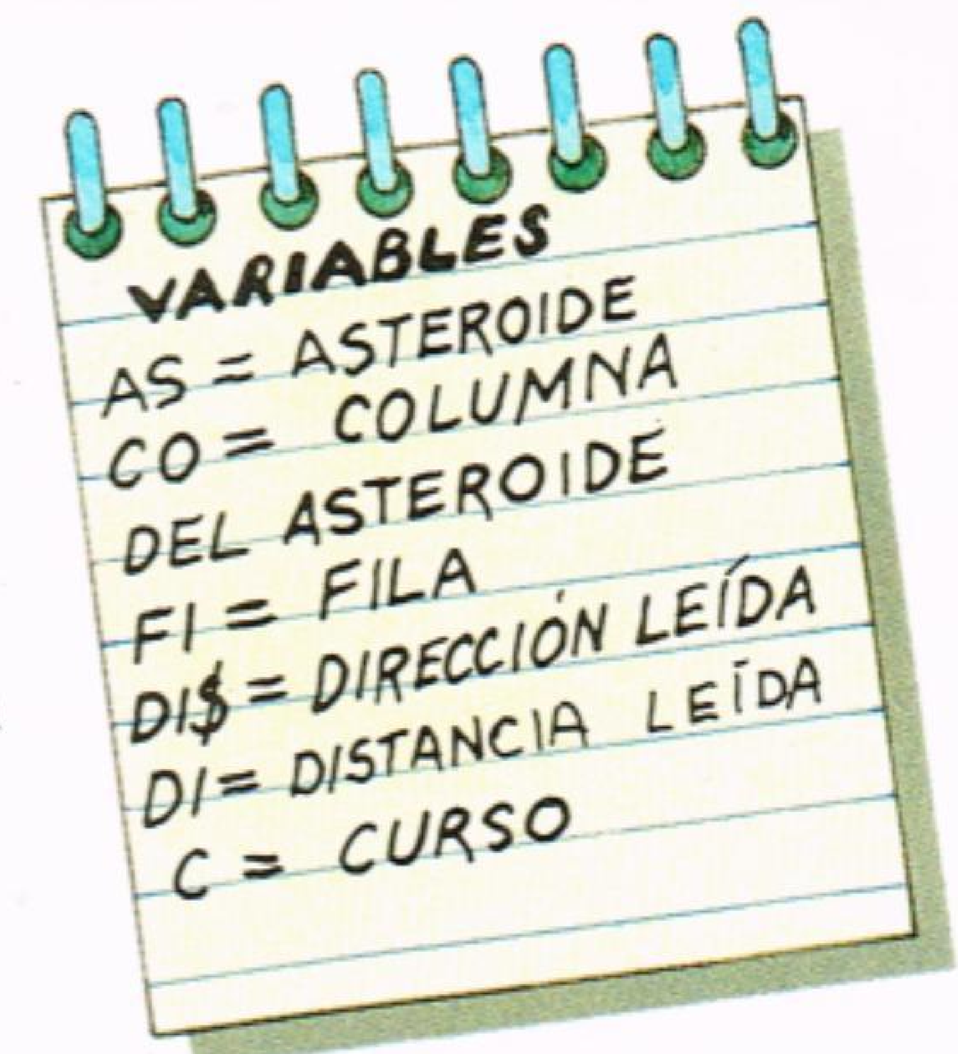


```
160 INPUT "DIRECCION (NORTE,SUR,ESTE) ";DI$
170 IF DI$="N" THEN AR=-1:DE=0:GOTO 1210
180 IF DI$="S" THEN AR=1:DE=0:GOTO 1210
190 IF DI$="E" THEN AR=0:DE=1:GOTO 1210
200 GOTO 1120
210 INPUT"DISTANCIA";DI
220 PRINTLEFT$(UD$,FI)SPC(CO)"0"
```

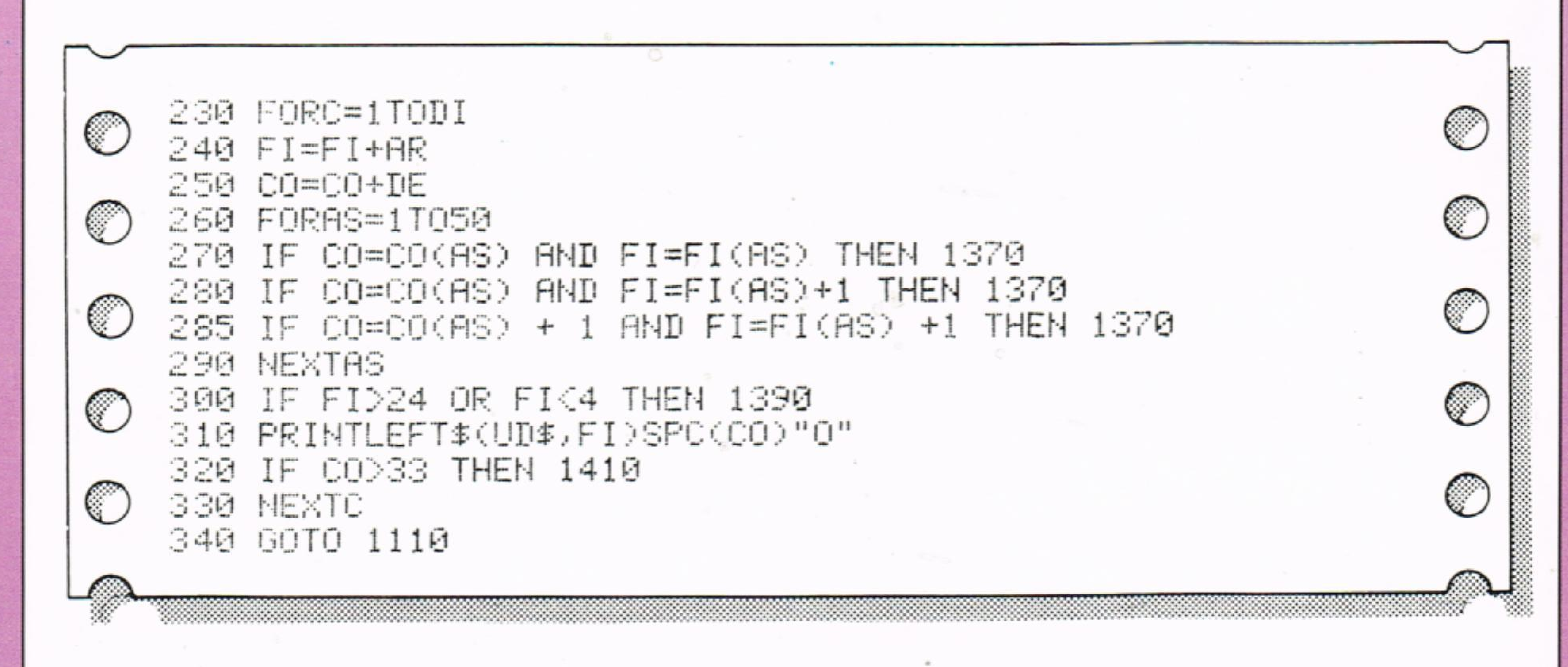
Las líneas 160 a 220 permiten la introducción de la dirección y la distancia del vuelo de la nave. Son suficientes tres direcciones —norte, sur y este—, ya que la nave se mueve de izquierda a derecha. La entrada se almacena en la variable **DI$**. Si se pulsa una tecla incorrecta la línea 200 manda al programa a la 120, hasta que se pulse una tecla válida. En la línea 220 el símbolo de la nave se cambia por una "0", para representar la trayectoria seguida.



Las líneas 170 a 190 cargan los valores para cambiar el curso de la nave, utilizando **AR** para los cambios en vertical y **DE** para los horizontales. El movimiento de la nave tiene lugar tras haber introducido la distancia. Esto se produce en la línea 210. Observa que no se necesita una variable alfanumérica para este **INPUT**, pues la distancia es un número. Por último, en esta sección el símbolo de la nave se cambia por un 0, para que el camino recorrido por la nave y la posición de ésta no se confundan.



Las líneas 230 a 330 son realmente el corazón del programa. Hacen que la nave se mueva por la pantalla y comprueban que no ha habido ningún choque. Toda la sección está dentro de un bucle **FOR...NEXT**, que usa la variable **C** (indicando el curso). El bucle se repite tantas veces como indica el valor de la distancia leído en la línea 210. La nueva posición de la nave se calcula en las líneas 240 y 250, utilizando los valores de **AR** y **DE** hallados en la sección anterior. La nave se mueve a su nueva fila y columna paso a paso.

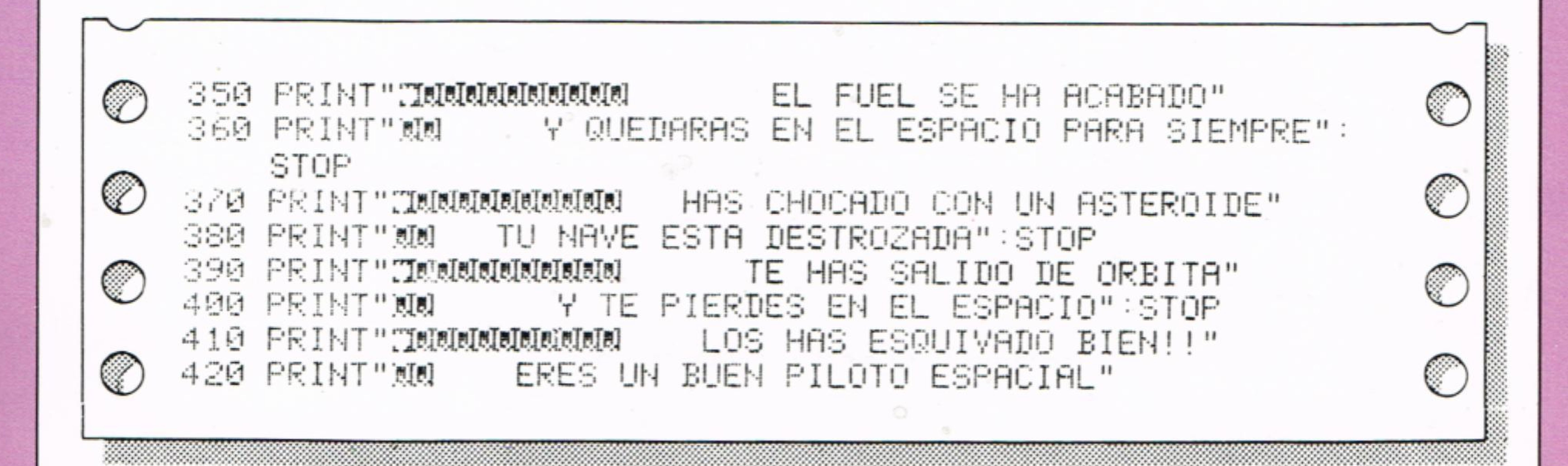


```
230 FORC=1TODI
240 FI=FI+AR
250 CO=CO+DE
260 FORAS=1TO50
270 IF CO=CO(AS) AND FI=FI(AS) THEN 1370
280 IF CO=CO(AS) AND FI=FI(AS)+1 THEN 1370
285 IF CO=CO(AS) + 1 AND FI=FI(AS) +1 THEN 1370
290 NEXTAS
300 IF FI>24 OR FI<4 THEN 1390
310 PRINTLEFT$(UD$,FI)SPC(CO)"O"
320 IF CO>33 THEN 1410
330 NEXTC
340 GOTO 1110
```



En las líneas 260 a 290 se usa otro bucle **FOR...NEXT** para comprobar si la nave ha chocado con algún asteroide. Las líneas 270 a 285 comparan la columna y la fila de la nave con las de los asteroides, almacenadas en las matrices creadas en la línea 20. La línea 270 comprueba la colisión con el carácter gráfico de la línea 60, y las líneas 280 y 285 con los de la 65. El bucle realiza todo esto para cada uno de los asteroides. La línea 300 comprueba si la nave se ha salido de órbita. Si todo ha ido bien, la nave se dibuja en su nueva posición. La comprobación final consiste en ver si la nave ha llegado a la columna 34. Por último, la línea 340 hace saltar el programa a la línea 110, para leer una nueva dirección y una nueva distancia.

Sólo queda añadir los mensajes finales. Los códigos de control de cursor y la colocación de mensajes en dos líneas producen una pantalla mejor. El símbolo del corazón en inverso delante de cada mensaje es el código para borrar la pantalla (hace lo mismo que **CHR$(147)**).



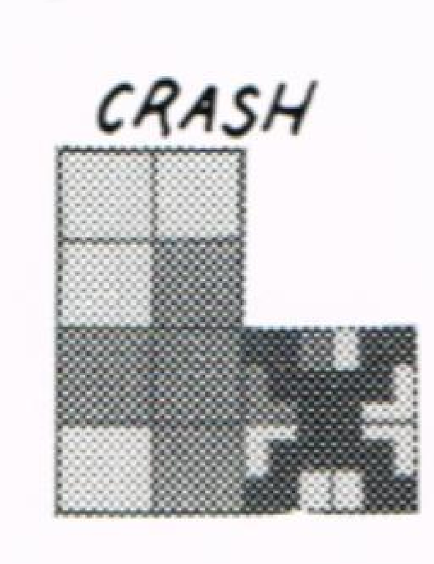
```
350 PRINT"           EL FUEL SE HA ACABADO"
360 PRINT"      Y QUEDARAS EN EL ESPACIO PARA SIEMPRE":
    STOP
370 PRINT"         HAS CHOCADO CON UN ASTEROIDE"
380 PRINT"     TU NAVE ESTA DESTROZADA":STOP
390 PRINT"          TE HAS SALIDO DE ORBITA"
400 PRINT"       Y TE PIERDES EN EL ESPACIO":STOP
410 PRINT"         LOS HAS ESQUIVADO BIEN!!"
420 PRINT"      ERES UN BUEN PILOTO ESPACIAL"
```

Al jugar debes tener en mente cómo están hechos los asteroides. Están compuestos por tres caracteres gráficos. Los caracteres están compuestos por cuatro bloques, como se indica en el diagrama de abajo. Si la nave se sitúa en cualquiera de los bloques de un carácter, el ordenador informará que se ha producido una colisión.

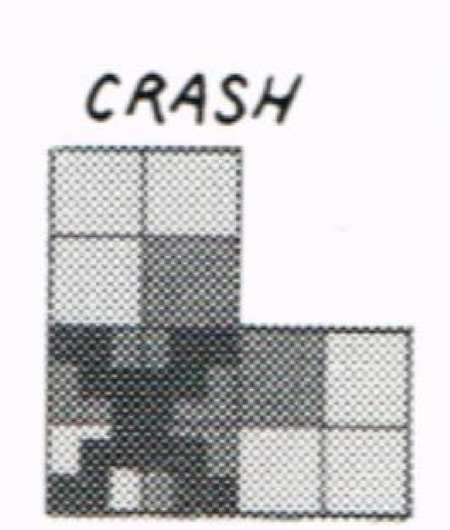
EVITADO



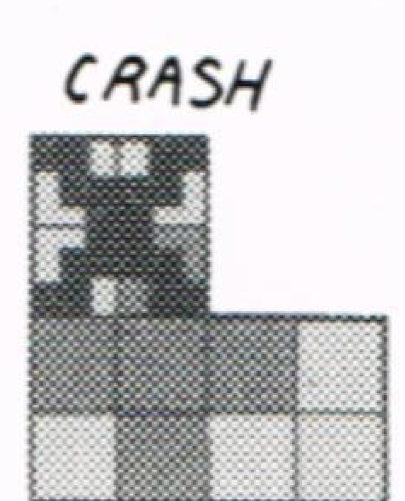
CRASH



CRASH



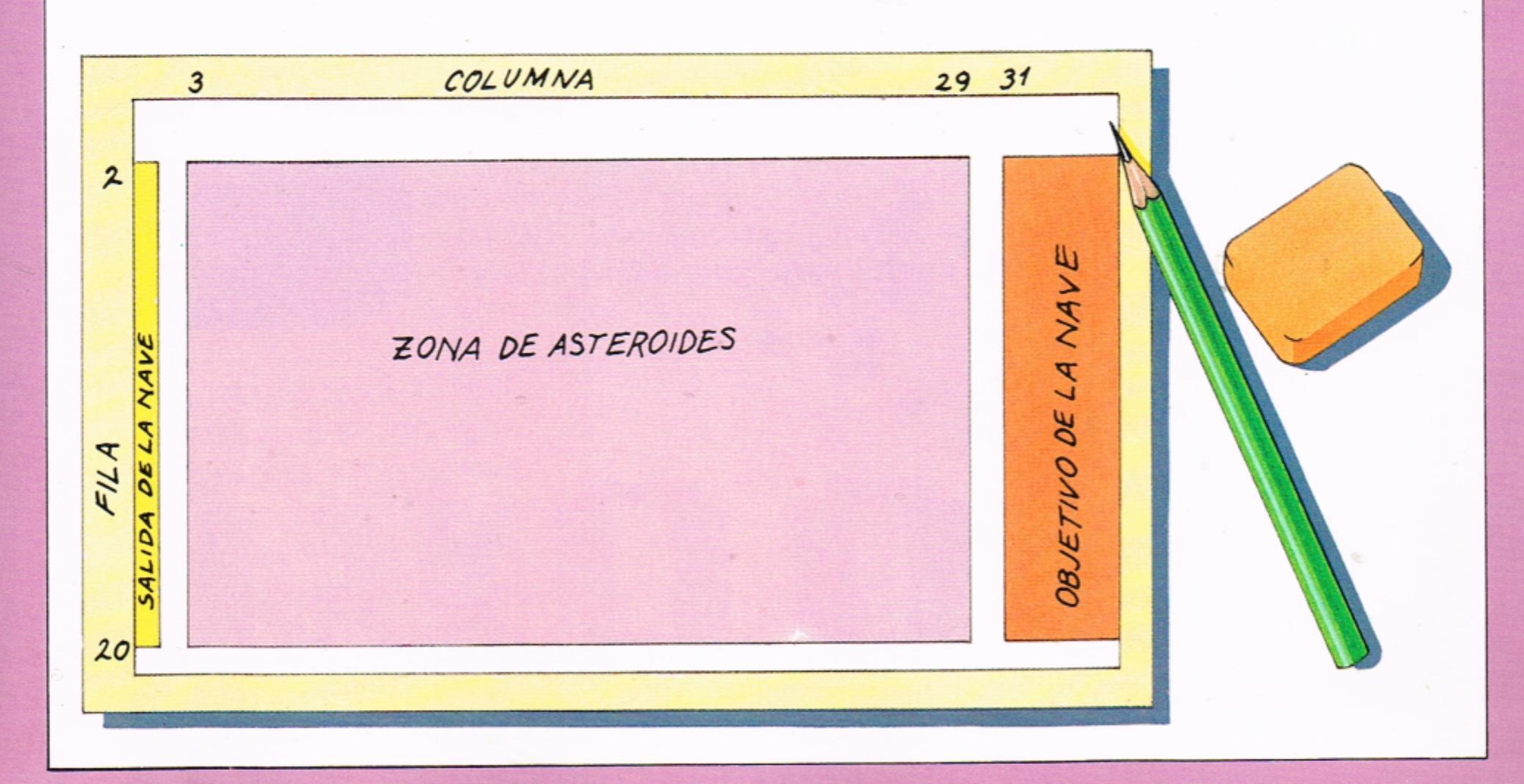
CRASH



Este programa usa el conjunto de caracteres gráficos disponibles en el Spectrum. Imprime los asteroides y el cohete en la pantalla y permite al cohete moverse de acuerdo con las órdenes que se le den durante el juego. Como anteriormente, una línea **REM** asigna un título al programa, y se limpia la pantalla de cualquier mensaje que hubiera.

```
 5 REM asteroides
10 CLS
```

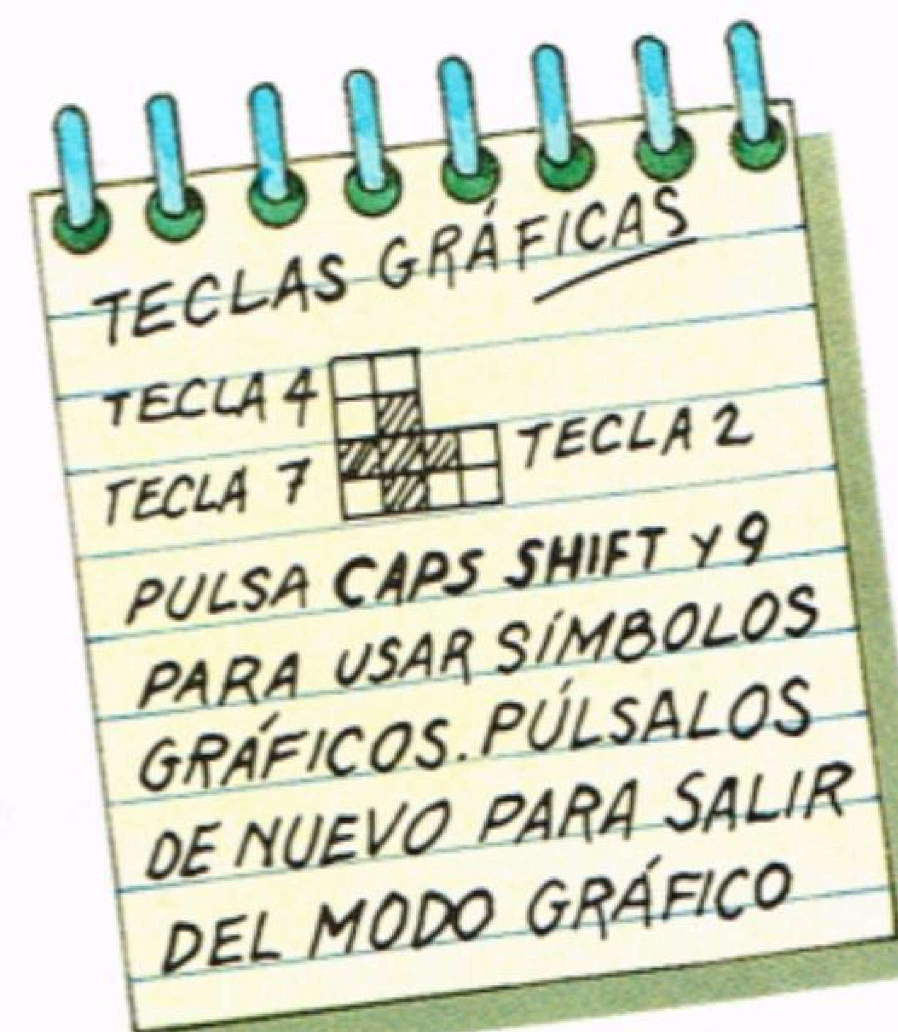
Lo primero, antes de escribir la sección del programa que dibujará los asteroides en posiciones aleatorias de la pantalla, es planear cómo va a quedar la pantalla. La pantalla del Spectrum tiene una anchura de 32 columnas y una altura de 22 filas, numeradas desde 0 a 31 y desde 0 a 21 respectivamente. El cohete debe atravesar la pantalla de izquierda a derecha; por tanto, su punto de salida debe estar en la columna 0. El campo de asteroides llegará hasta la columna 29 y la nave conseguirá su objetivo cuando pase la columna 30 sin haber chocado con ningún asteroide. Las dos últimas filas de la pantalla se reservarán para imprimir instrucciones y mensajes indicando el desarrollo del juego y para permitir la introducción de órdenes durante el juego. Además, la nave no podrá volar por el borde de la pantalla para esquivar los asteroides. Teniendo en cuenta estas consideraciones, se hace el planteamiento indicado más abajo. Puedes hacer otra planificación diferente, pero si lo haces deberás cambiar algunos de los valores de las instrucciones **PRINT AT** que hay en el programa para posicionarse en un cuadro determinado. Dibuja una cuadrícula de la pantalla para ayudarte a organizar la impresión.

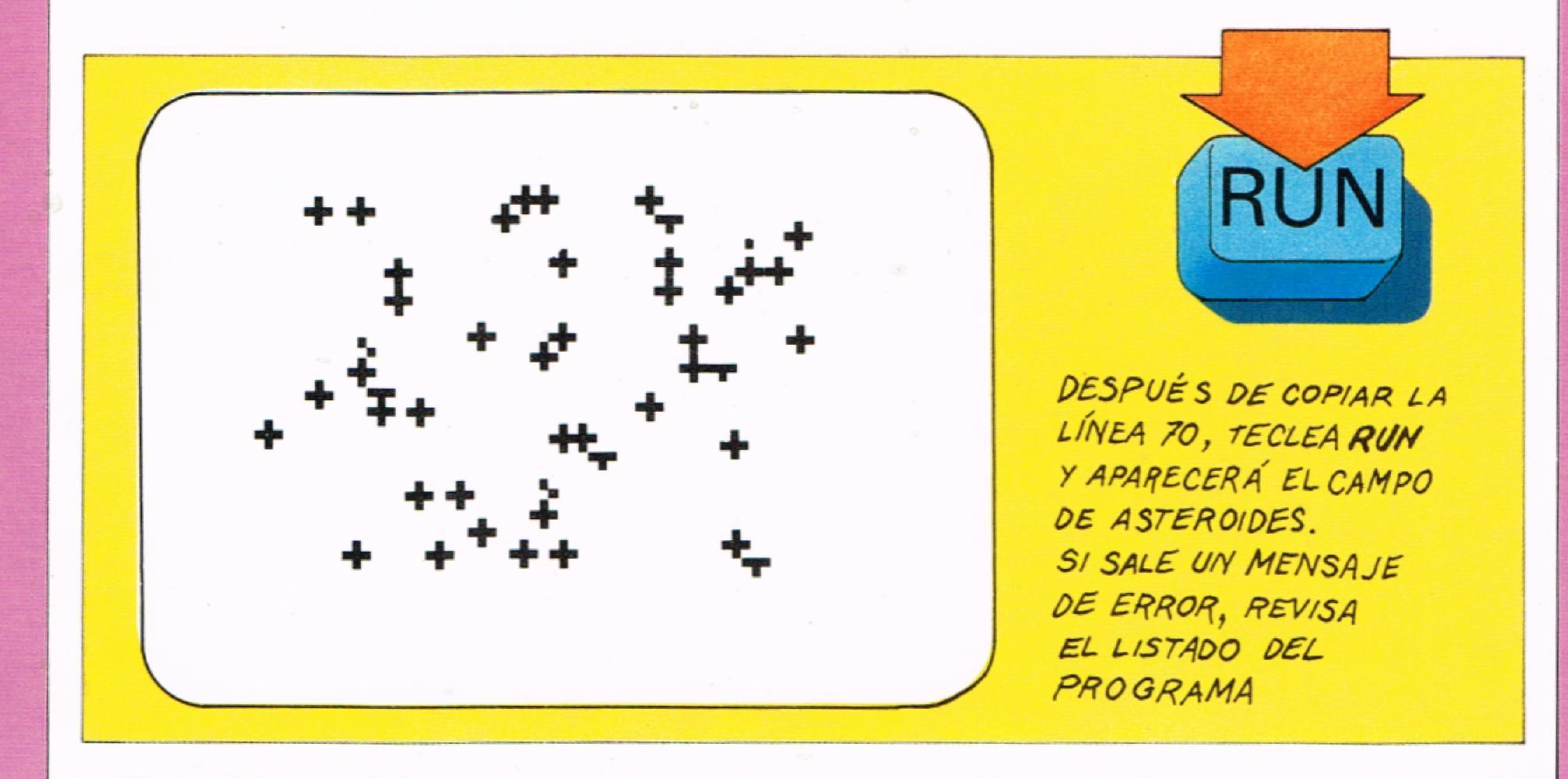


```
20 DIM c(50): DIM f(50)
30 FOR s=1 TO 50
40 LET c(s)=INT (RND*26)+3
50 LET f(s)=INT (RND*18)+2
60 PRINT AT f(s),c(s); CHR$ 132
65 PRINT AT f(s)+1,c(s); CHR$ 135;CHR$ 130
70 NEXT s
```

En la línea 20, la orden **DIM** crea dos matrices, donde se van a guardar las coordenadas columna y fila de los asteroides. El número entre paréntesis indica el número de asteroides y por tanto el tamaño necesario para la matriz. Las líneas 30 a 70 son un bucle **FOR...NEXT** que hace que las órdenes de las líneas 40 a 65 se realicen para cada uno de los 50 asteroides. La variable del bucle **"s"** indica el asteroide que toca; en el Spectrum, las variables de bucle constan de una sola letra. Al final del programa tienes una lista con todas las variables utilizadas. Es una buena idea hacer una lista de ellas, pues resulta fácil olvidar para qué sirve cada una. En las líneas 40 a 50 se generan una columna y una fila aleatorias que indican la posición de cada asteroide en la pantalla. Las líneas 60 y 65 imprimen los gráficos que representan los asteroides; están formados por tres gráficos con códigos **CHR$** de **132**, **135** y **130.** Estos gráficos están también disponibles directamente en el teclado (ver la nota adjunta).

Si ejecutas esta sección verás cómo se genera el campo de asteroides en posiciones aleatorias de la pantalla.





Una vez dibujado el campo de asteroides, hay que dar a la nave una posición de salida aleatoria en la columna 0. La instrucción **RND** nos sirve de nuevo, y el valor generado lo convertimos en un entero entre dos y veinte. La nave se representa con una X. La próxima sección del programa permite al jugador controlar su movimiento por la pantalla.

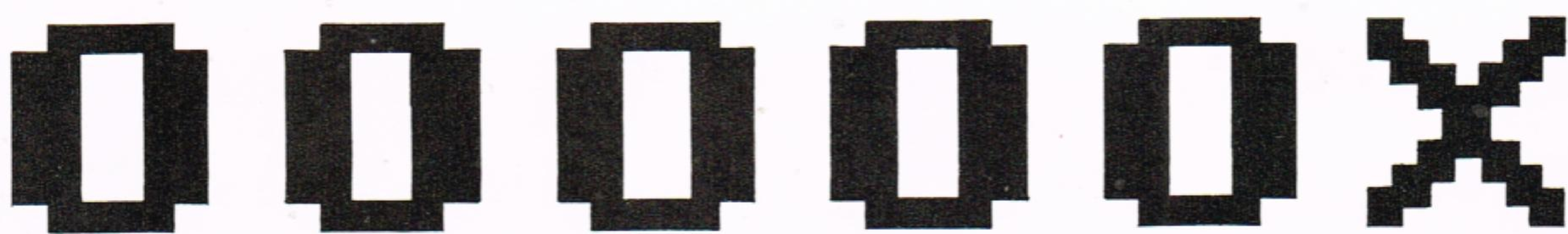
```
100 LET col=0: LET fila=2+INT (RND*18)
110 PRINT AT fila,col;"X"
```

Para controlar la nave son suficientes tres direcciones —norte, sur y este—, pues la nave atraviesa la pantalla de izquierda a derecha. La dirección se introduce usando las letras N, S y E respectivamente. Con **INPUT** la almacenamos en la cadena **d$**. Si se introduce una letra incorrecta, la línea 200 manda al programa a la línea 160, y así ocurrirá hasta que se teclee una letra correcta y el programa salte a la línea 210.

```
160 INPUT "DIRECCION (norte,sur,este) ";d$
170 IF d$="n" THEN  LET arriba=-1: LET derecha=0: GO TO 210
180 IF d$="s" THEN  LET arriba=1: LET derecha=0: GO TO 210
190 IF d$="e" THEN  LET arriba=0: LET derecha=1: GO TO 210
200 GO TO 160
210 INPUT "DISTANCIA ";distancia
220 PRINT AT fila,col;"O"
```

Las líneas 170 a 190 cargan los valores para cambiar el curso de la nave, utilizando **arriba** para los cambios en vertical y **derecha** para los horizontales. El movimiento de la nave tiene lugar tras haber introducido la distancia. Esto se produce en la línea 210. Observa que no se necesita una variable alfanumérica para este **INPUT**, pues la distancia es un número. Por último, en esta sección el símbolo de la nave se cambia por una 0, para que el camino recorrido por la nave y la posición de ésta no se confundan.

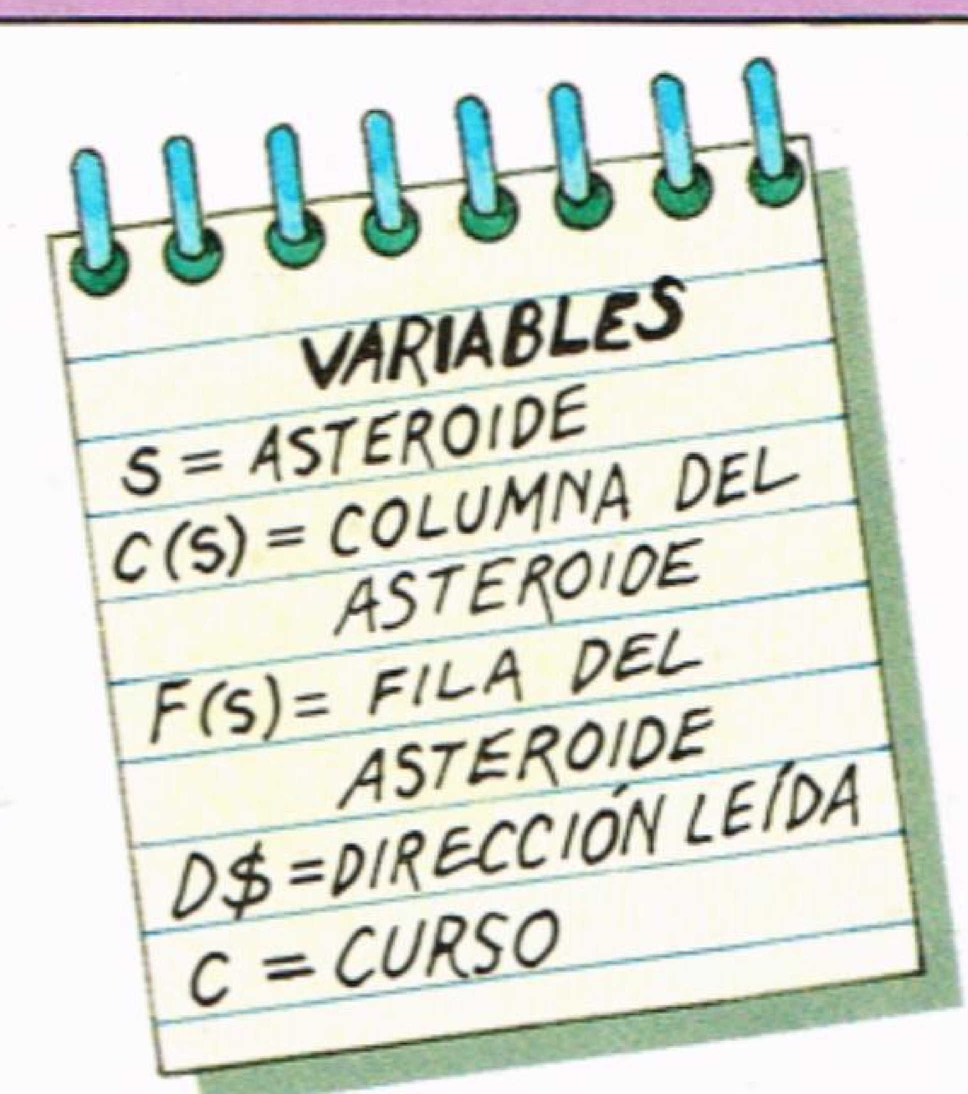


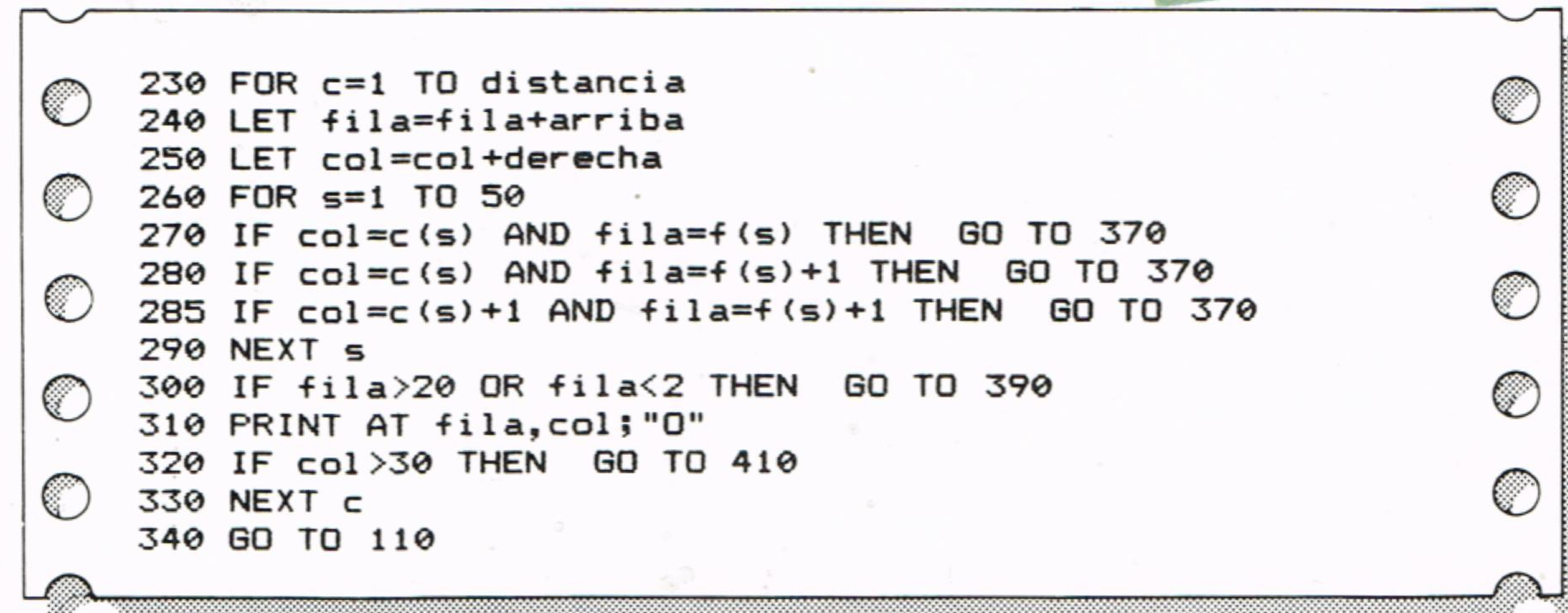


CAMINO DE LA NAVE

POSICIÓN DE LA NAVE

Las líneas 230 a 330 son realmente el corazón del programa. Hacen que la nave se mueva por la pantalla y comprueban que no ha habido ningún choque. Toda la sección está dentro de un bucle **FOR...NEXT**, que usa la variable **c.** El bucle se repite tantas veces como indica el valor de distancia leído en la línea 210. La nueva posición de la nave se calcula en las líneas 240 y 250, usando los valores de **arriba** y **derecha** hallados en la sección anterior. La nave se mueve a su nueva fila y columna paso a paso, imprimiendo un "0" por su camino.





```
230 FOR c=1 TO distancia
240 LET fila=fila+arriba
250 LET col=col+derecha
260 FOR s=1 TO 50
270 IF col=c(s) AND fila=f(s) THEN  GO TO 370
280 IF col=c(s) AND fila=f(s)+1 THEN  GO TO 370
285 IF col=c(s)+1 AND fila=f(s)+1 THEN  GO TO 370
290 NEXT s
300 IF fila>20 OR fila<2 THEN  GO TO 390
310 PRINT AT fila,col;"O"
320 IF col>30 THEN  GO TO 410
330 NEXT c
340 GO TO 110
```

En las líneas 260 a 290 se usa otro bucle **FOR...NEXT** para comprobar si la nave ha chocado con algún asteroide. Las líneas 270 a 285 cogen la columna y la fila de la nave y las comparan con las de los asteroides, almacenadas en las matrices creadas en la línea 20. La línea 270 comprueba la colisión con el carácter gráfico de la línea 60, y las líneas 280 y 285 con los de la 65. El bucle realiza todo esto para cada uno de los asteroides. La línea 300 comprueba si la nave se ha salido de órbita. Si todo ha ido bien, la nave se dibuja en su nueva posición. La comprobación final consiste en ver si la nave ha llegado a la columna 30. Por último, la línea 340 hace saltar el programa a la línea 110, para leer una nueva dirección y una nueva distancia.

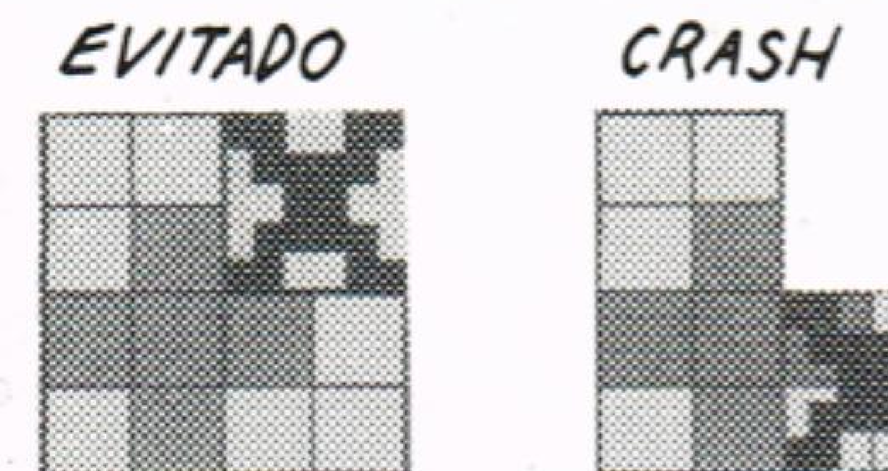


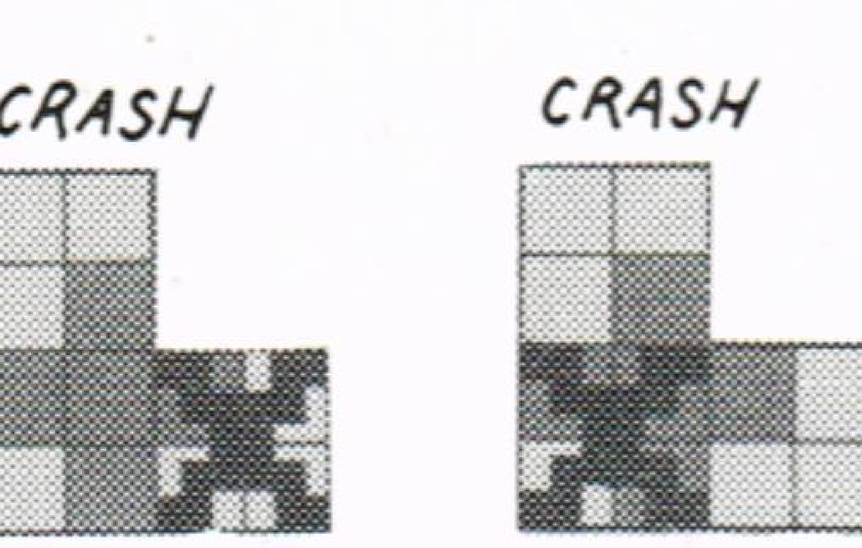
Sólo queda añadir los mensajes finales. Las instrucciones **PRINT AT** y la colocación de mensajes en dos líneas producen una pantalla mejor. No olvides las órdenes **STOP**; si omites alguna, el ordenador seguirá hasta la última línea del programa.

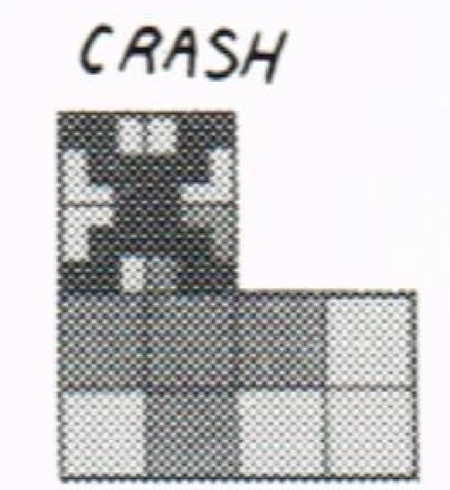
```
370 CLS : PRINT AT 10,0;"HAS CHOCADO CON UN ASTEROIDE"
380 PRINT AT 13,0;"TU NAVE ESTA DESTROZADA": STOP
390 CLS : PRINT AT 10,0;"TE HAS SALIDO DE ORBITA"
400 PRINT AT 13,0;"Y TE PIERDES EN EL ESPACIO": STOP
410 CLS : PRINT AT 10,0;"LOS HAS ESQUIVADO BIEN !!"
420 PRINT AT 20,0;"ERES UN BUEN PILOTO ESPACIAL"
```

DIRECCION (norte, sur, este) "n"

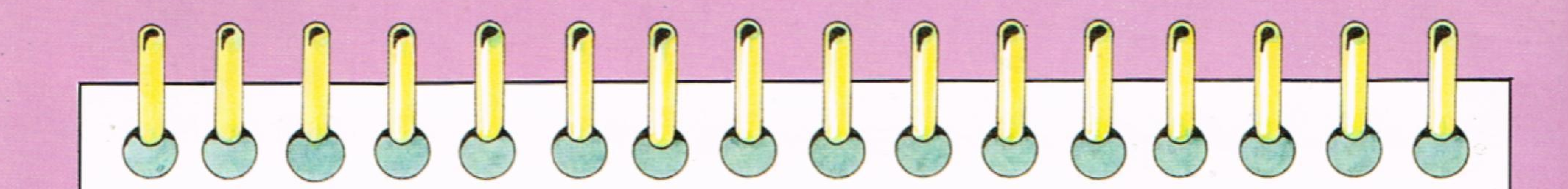
Al jugar debes tener en mente cómo son los asteroides. Están compuestos por tres caracteres gráficos. En el Spectrum, cada carácter gráfico disponible está formado por una combinación de cuatro bloques, como se muestra en el diagrama de abajo. El símbolo de la nave también ocupa cuatro bloques. Si cualquier parte de la nave ocupa alguno de los cuatro bloques de alguno de los tres caracteres gráficos, el ordenador informará que se ha producido una colisión.







CRASH



Mejora el programa

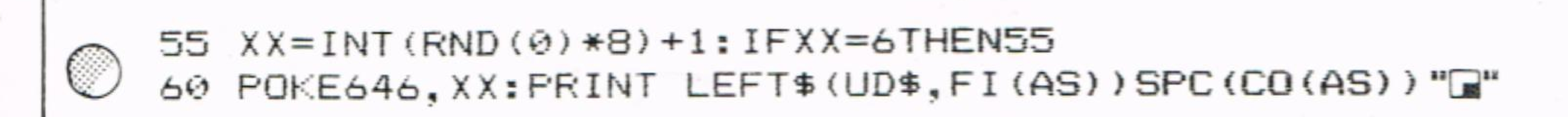
Puedes hacer el juego más difícil incrementando el número de asteroides —con unos 120 es casi imposible encontrar una ruta libre. Cambia las líneas 20, 30 y 260.



Adición de color

COMMODORE

Para producir asteroides de color aleatorio en el Commodore se debe usar la instrucción POKE. La dirección de memoria del ordenador 646 selecciona el color. El segundo número que sigue a POKE indica el color elegido. La sección de programa que sigue elige un número aleatorio entre 1 y 8, pero este valor no puede ser seis, pues es el color del papel y produciría un asteroide "invisible".



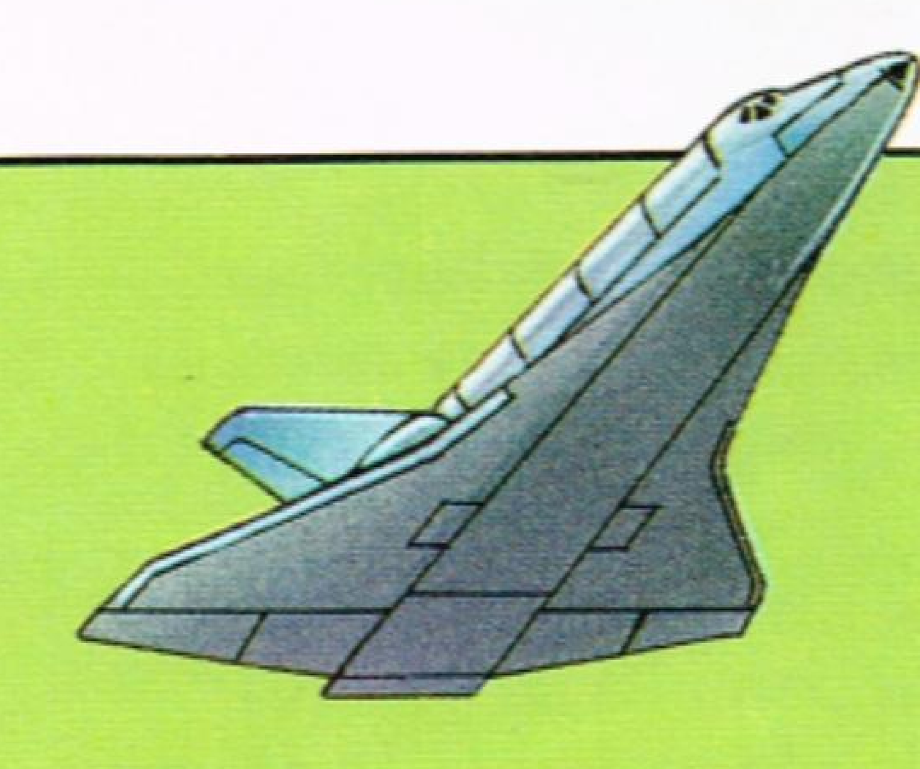
```
55 XX=INT(RND(0)*8)+1:IFXX=6THEN55
60 POKE646,XX:PRINT LEFT$(UD$,FI(AS))SPC(CO(AS))"▣"
```

SPECTRUM

Es muy fácil hacer los asteroides de colores diferentes. El Spectrum puede producir asteroides de colores usando la orden **INK**, seguida de un número entre 0 y 7, ambos inclusive (mira las teclas del color del teclado y consulta tu manual). Puedes utilizar la capacidad del ordenador para generar números aleatorios dentro de cualquier rango, en nuestro caso de 0 a 7, añadiendo la línea 55 y cambiando la línea 60 como se indica. Así producirás asteroides de color aleatorio.

```
55 LET xxx=INT (RND*7)
60 PRINT AT f(s),c(s); INK xxx;CHR$ 132
65 PRINT AT f(s)+1,c(s); INK xxx;CHR$ 135;CHR$ 130
```

Usa el sonido para producir efectos al salirte de órbita o para cuando consigas atravesar el campo de asteroides.

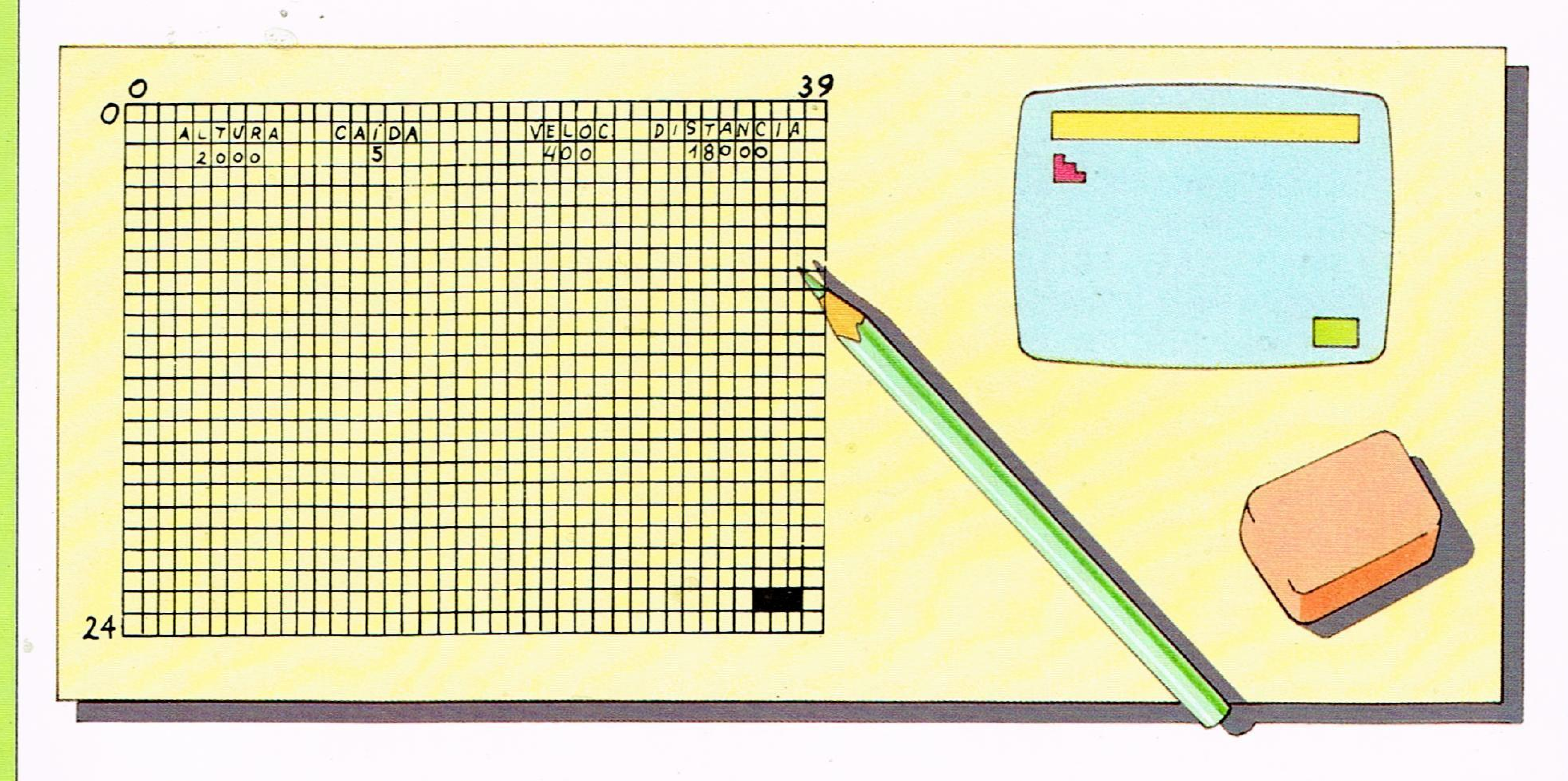


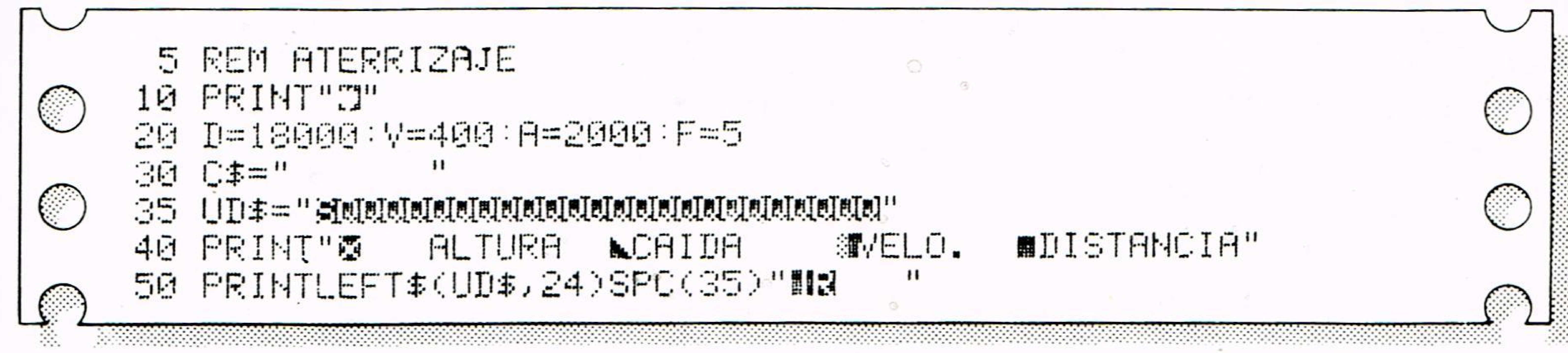
ATERRIZAJE DEL TRANSBORDADOR ESPACIAL

Por causa de la tormenta de asteroides, tu misión ha quedado suspendida, y debes regresar a la base. El punto de aterrizaje está a la vista, pero el piloto automático no funciona. Debes llevar la nave a tierra usando los dos controles manuales. Tu panel de control te dará detalles del vuelo.

ALTURA	CAIDA	VELO.	DISTANCIA
1200	5	400	11600

Como en el programa anterior, el primer paso es hacer un plano de cómo va a quedar la pantalla. De nuevo la nave va a atravesar la pantalla de izquierda a derecha, empezando en la esquina de arriba. El objetivo de la nave, la pista de aterrizaje, se representa con un bloque en la esquina de abajo de la derecha. La parte de arriba de la pantalla se reserva para el panel de control de la nave, donde se indica la altura, el grado de caída, la velocidad y la distancia a la pista. Con estas consideraciones, nos queda una pantalla como ésta.

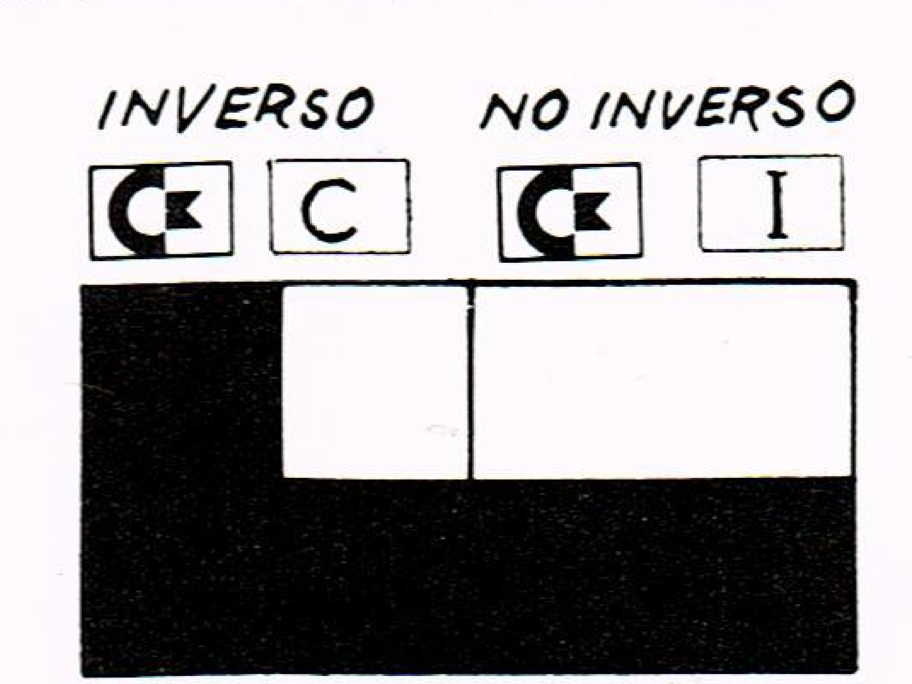




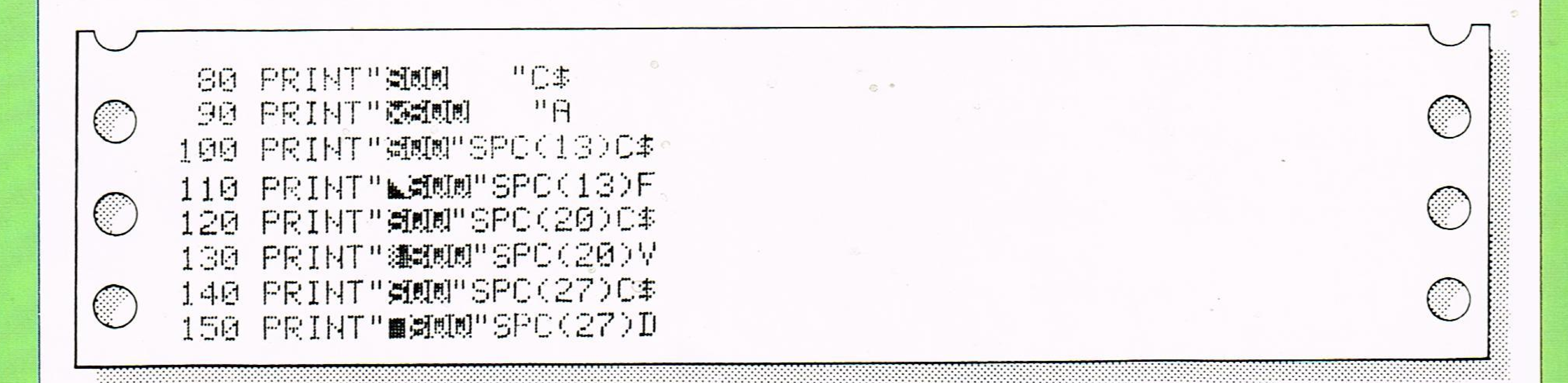
```
 5 REM ATERRIZAJE
10 PRINT"♥"
20 D=18000:V=400:A=2000:F=5
30 C$="      "
35 UD$="▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓"
40 PRINT"▓   ALTURA   ▓CAIDA     ▓VELO.   ▓DISTANCIA"
50 PRINTLEFT$(UD$,24)SPC(35)"▓▓    "
```

La línea 20 pone los valores iniciales para la velocidad, altura, grado de caída y distancia de la nave a la pista. En la línea 30 se almacenan seis espacios en la cadena **C$**, que utilizaremos luego para borrar de la pantalla los valores antiguos del panel de control. La línea 35 almacena un conjunto de códigos de control de cursor, que se usará más tarde para posicionar el cursor. El panel de control se imprime en la línea 40. El color del texto se consigue pulsando la tecla Control o la tecla Commodore junto con un número de 1 a 8. Por ejemplo, el rojo claro se obtiene con Commodore y "3". En el programa los colores están representados con códigos de color en las líneas 40 y 50. El código de la línea 50 produce en la pista un color verde claro.

```
60 X=0:Y=3
70 PRINTLEFT$(UD$,Y+1)SPC(X)"  "
```



La posición inicial de la nave se da por medio de las coordenadas X e Y de la línea 60. La nave se imprime en amarillo, usando los códigos de "inverso" y "no inverso", de forma similar a como se hizo con los asteroides en el programa anterior. La cola de la nave se hace con la tecla Commodore y "C" y el cuerpo con la "I".



```
 80 PRINT"       "C$
 90 PRINT"        "A
100 PRINT"   "SPC(13)C$
110 PRINT"    "SPC(13)F
120 PRINT"   "SPC(20)C$
130 PRINT"    "SPC(20)V
140 PRINT"   "SPC(27)C$
150 PRINT"    "SPC(27)D
```

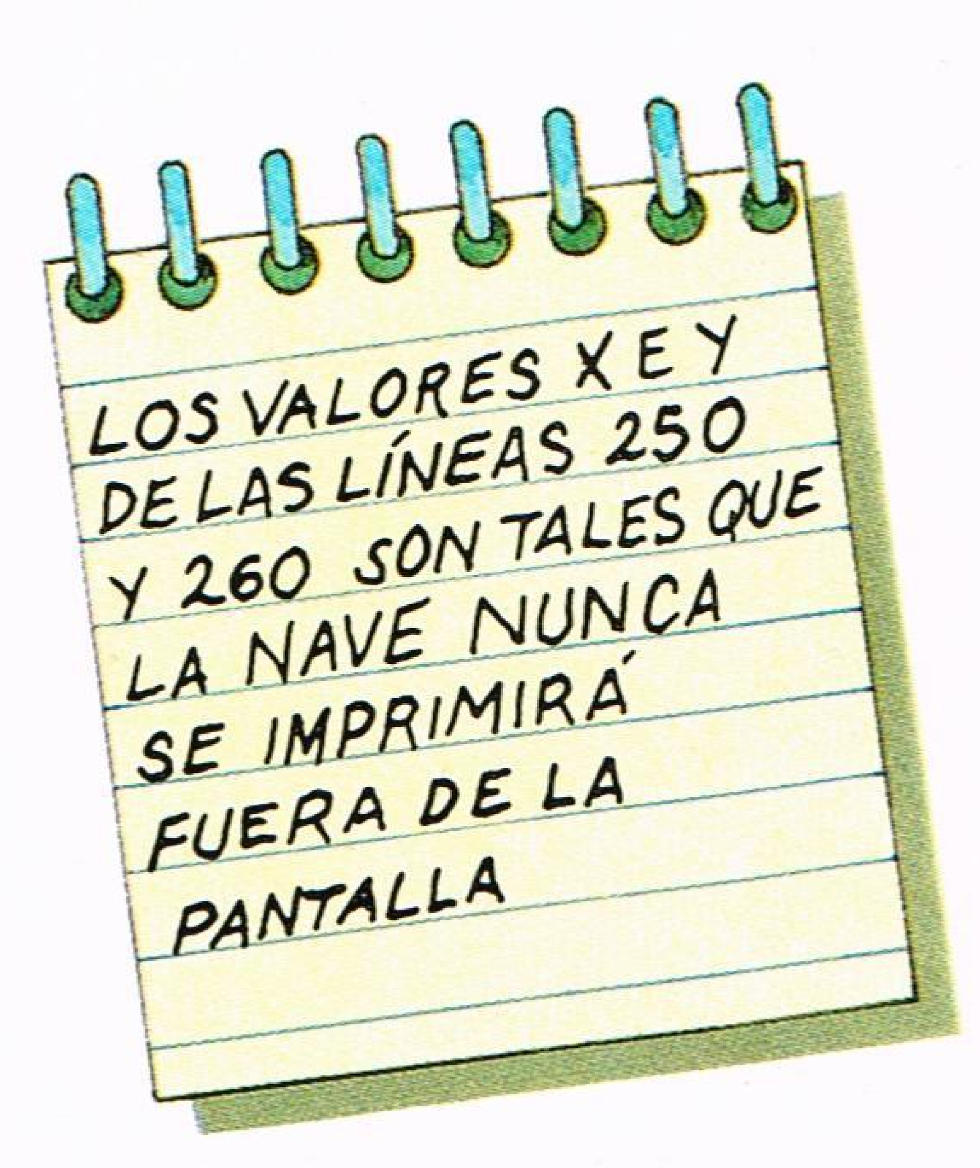
Las líneas 80 a 150 imprimen los valores de la velocidad, grado de caída, altura y distancia a la pista, cada uno debajo de su correspondiente cabecera en el panel de control. De nuevo se usan los códigos de control de color para producir un texto coloreado. La cadena **C$** de las líneas 80, 100, 120 y 140 utiliza los seis espacios almacenados en la línea 30 para borrar los antiguos valores antes de imprimir los nuevos. Esto dará la impresión de un panel de control en funcionamiento a medida que la nave se aproxime a la pista de aterrizaje.

```
160 IFV<200 THENPRINTLEFT$(UD$,11)SPC(12)" EL MOTOR SE HA
    PARADO":STOP
170 IF D<-999 THEN PRINTLEFT$(UD$,11)" EL COHETE SE HA
    PASADO":STOP
180 IFA<=0 THEN 2290
```

Las líneas 160 a 180 comprueban que la velocidad no sea demasiado baja, y que la nave no se haya pasado de la pista en un valor mayor de 999. La línea 180 comprueba si la nave ha llegado a una altura cero, en cuyo caso el programa salta a la línea 290, donde se comprueba si se cumplen las otras condiciones necesarias para el aterrizaje. La instrucción **LEFT$(UD$,11)** usa un trozo de la cadena de códigos de control de cursor, que se almacenaron en la línea 35, para posicionar los mensajes en la pantalla.

```
190 GET R$
200 IF R$="A" THEN F=F+1:V=V+5
210 IF R$="D" THEN F=F-1:V=V-5
220 IF F=0 THEN F=1:V=V-20
230 A=A-F:V=V+(F-5)*3:D=D-INT(V/10)
240 PRINTLEFT$(UD$,Y+1)SPC(X)C$
250 X=35-INT(D/(18000/35))
260 Y=22-INT(A/(2000/19))
270 PRINTLEFT$(UD$,Y+1)SPC(X)"■■ ■■"
280 GOTO 2080
```

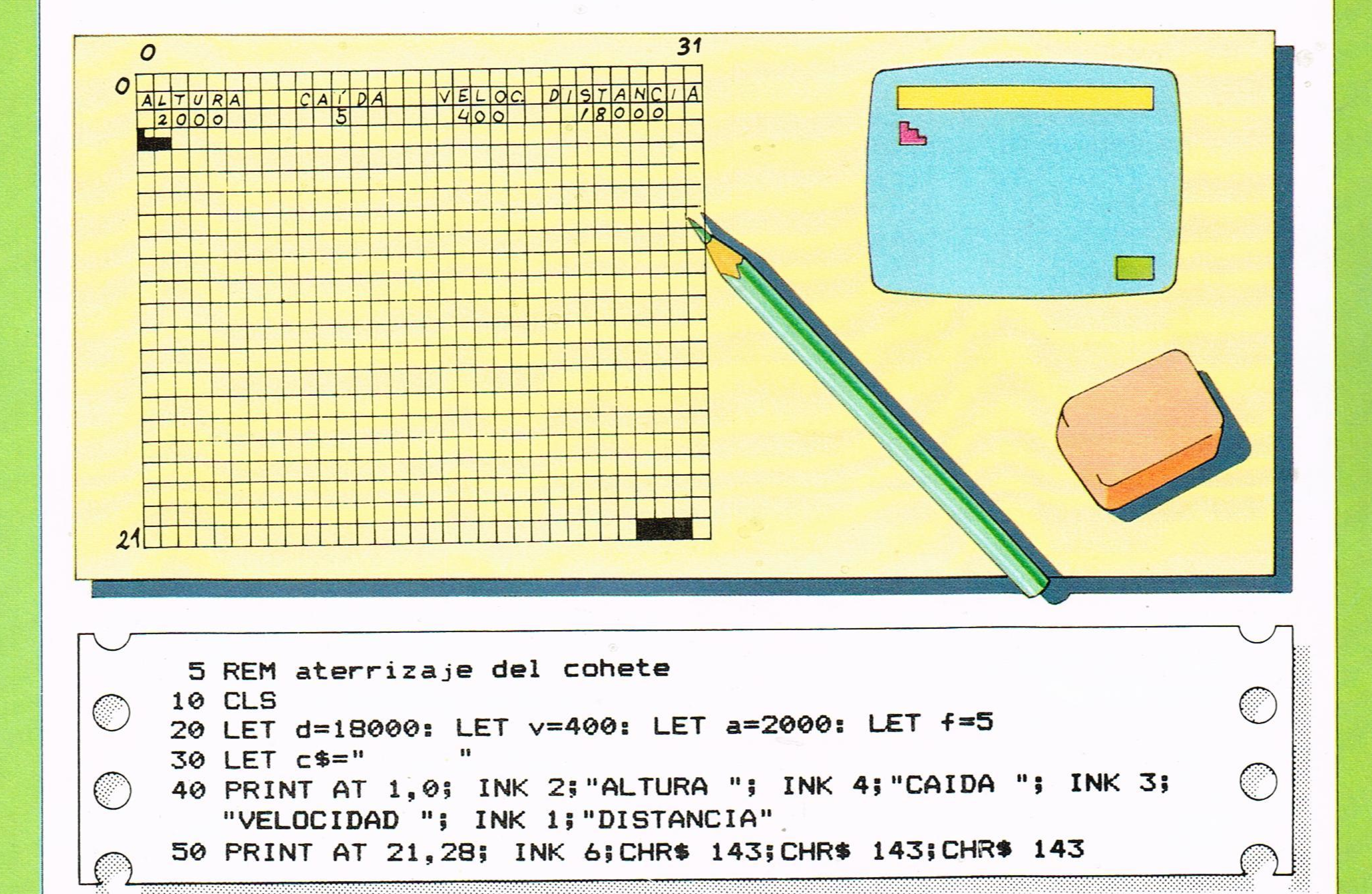
Desde la línea 190 hasta la 280, el programa permite al jugador controlar el descenso de la nave. En la línea 190, la orden **GET R$** le dice al ordenador que examine el teclado y vea si se ha pulsado alguna tecla. Cualquier tecla que se pulse se almacena en **R$**. Las líneas 200 y 210 comprueban si en **R$** hay una "A" o una "D" (que significan acelerar y decelerar respectivamente), en cuyo caso se modifican los valores de los indicadores de acuerdo con la tecla pulsada. En la línea 220 el programa se asegura de que el grado de descenso nunca sea cero (si así fuera, la nave se pararía del todo). La línea 230 calcula los nuevos valores de altura, velocidad y distancia, para ser imprimidos en el panel de control. En la línea 240, parte de los códigos de control de **UD$** se usan para posicionar el cursor, de forma que con los espacios almacenados en **C$** se borre la antigua posición de la nave. Los nuevos valores de las coordenadas X e Y se calculan en las líneas 250 y 260, usando los nuevos valores de distancia y altura. La línea 270 dibuja la nave en su nueva posición, y la 280 vuelve a la línea 80 para continuar con el descenso de la nave.




```
290 IF D>1000 THEN PRINT LEFT$(UD$,11)"■       ATERRIZAJE
    DEFECTUOS0":END
300 IF V>300 THEN PRINTLEFT$(UD$,11)"■       DEMASIADA
    VELOCIDAD!!":END
310 IF F>5THEN PRINTLEFT$(UD$,11)"■       TREN DE
    ATERRIZAJE ROTO":END
320 PRINTLEFT$(UD$,11)"■     ATERRIZAJE PERFECTO!!!":END
```

Por último vienen los mensajes de finalización. Se comprueba que el aterrizaje no sea demasiado lejos de la pista, y que ni la velocidad ni el grado de caída sean demasiado altos. Si se cumplen todas las condiciones, el programa va a la línea 320 y aparece el mensaje de felicitación.

Como en el progama anterior, el primer paso es hacer un plano de cómo va a quedar la pantalla. De nuevo la nave va a atravesar la pantalla de izquierda a derecha, empezando por la esquina de arriba. El objetivo de la nave, la pista de aterrizaje, se representa con un bloque en la esquina de abajo en la derecha. La parte de arriba de la pantalla se reserva para el panel de control de la nave, donde se indica la altura, el grado de caída, la velocidad y la distancia a la pista. Con estas consideraciones, nos queda una pantalla como ésta:



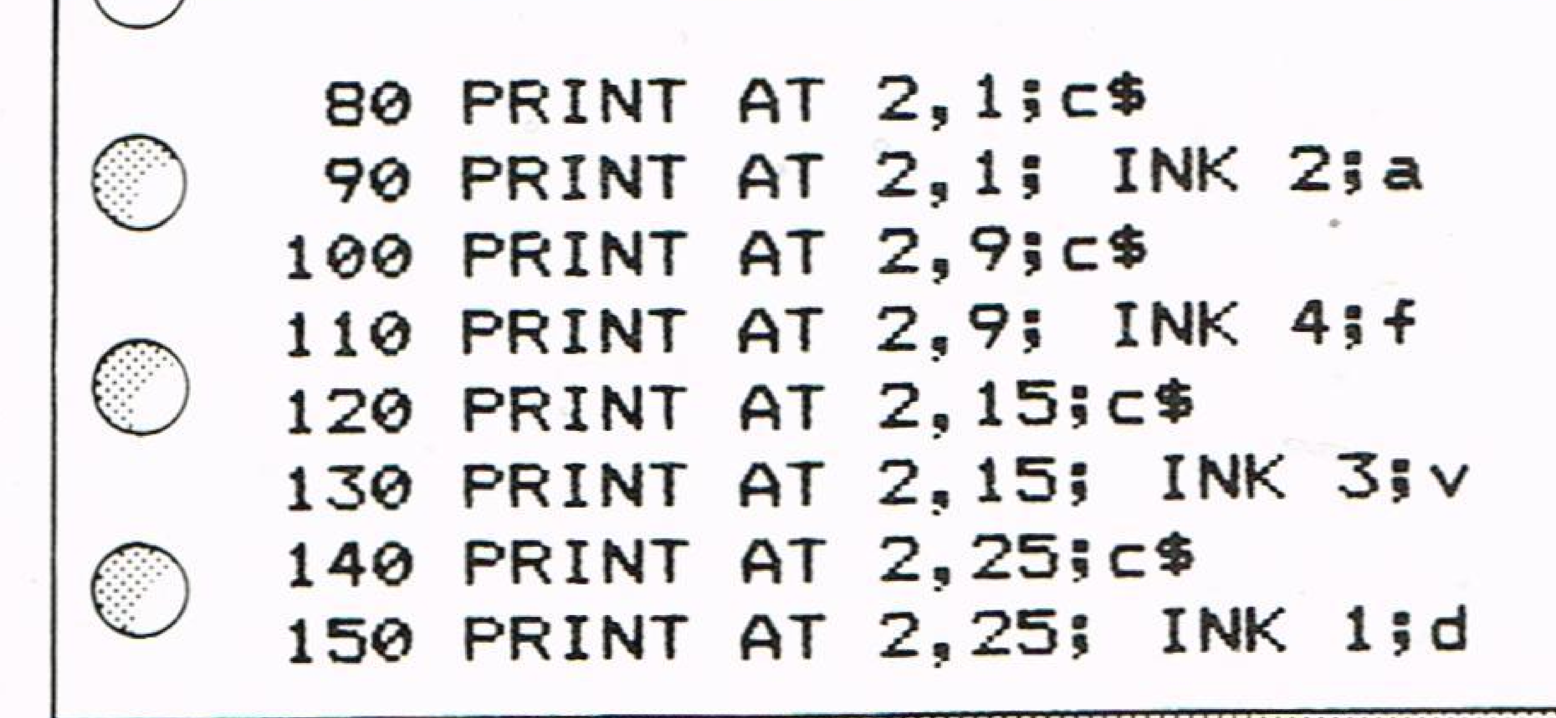

```
 5 REM aterrizaje del cohete
10 CLS
20 LET d=18000: LET v=400: LET a=2000: LET f=5
30 LET c$="     "
40 PRINT AT 1,0; INK 2;"ALTURA "; INK 4;"CAIDA "; INK 3;
   "VELOCIDAD "; INK 1;"DISTANCIA"
50 PRINT AT 21,28; INK 6;CHR$ 143;CHR$ 143;CHR$ 143
```

La línea 20 pone los valores iniciales para la velocidad, altura, grado de caída y distancia de la nave a la pista. En la línea 30 se almacenan cinco espacios en la cadena **c$**, que utilizaremos luego para borrar de la pantalla los valores antiguos del panel de control. El panel de control se imprime en la línea 40. Las órdenes **INK** producen diferentes colores en el panel. No olvides colocar el espacio detrás de cada palabra; es para que la línea quede colocada adecuadamente en la pantalla. Por último, la pista de aterrizaje se imprime en la línea 50, usando la orden **INK** para obtener color y con tres **CHR$143** para formar un bloque sólido. Como antes, puedes usar los caracteres gráficos directamente, en lugar de los códigos **CHR$**.

```
60 LET x=0: LET y=3
70 PRINT AT y,x; INK 1;CHR$ 142;CHR$ 140
```

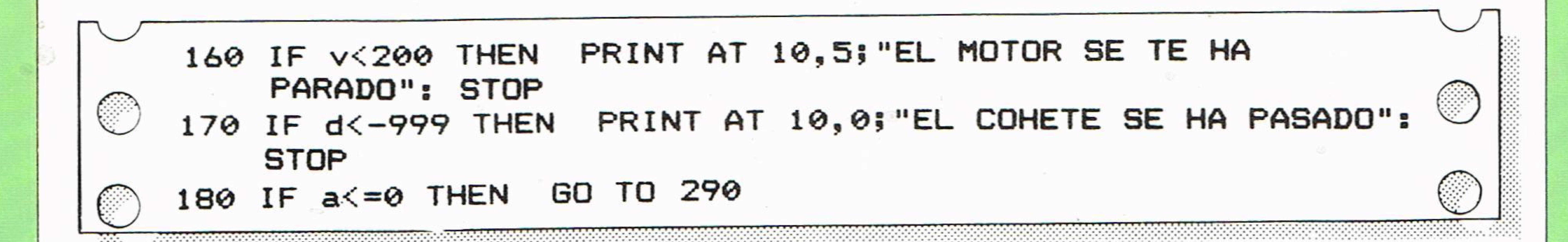


La posición de la nave en la pantalla se da por medio de las coordenadas X e Y de la línea 60, que indican la columna y la fila en que se encuentran. (Recuerda que la fila viene primero en las órdenes **PRINT AT**.) La nave se representa con bloques gráficos obtenidos con **CHR$**, coloreados con **INK.**



```
 80 PRINT AT 2,1;c$
 90 PRINT AT 2,1; INK 2;a
100 PRINT AT 2,9;c$
110 PRINT AT 2,9; INK 4;f
120 PRINT AT 2,15;c$
130 PRINT AT 2,15; INK 3;v
140 PRINT AT 2,25;c$
150 PRINT AT 2,25; INK 1;d
```

Las líneas 80 a 150 imprimen los valores de la velocidad, grado de caída, altura y distancia a la pista, usando la orden **PRINT AT** para colocarlos debajo de su correspondiente cabecera en el panel de control. De nuevo se usa **INK** para colorearlos. La cadena **c$** de las líneas 80, 100, 120 y 140 utiliza los cinco espacios almacenados en la línea 30 para borrar los antiguos valores antes de imprimir los nuevos. Dará la impresión de un panel de control en funcionamiento a medida que la nave se aproxime a la pista de aterrizaje. Este método de sobreimprimir partes de la pantalla se usa a menudo en los programas para producir gráficos en movimiento.

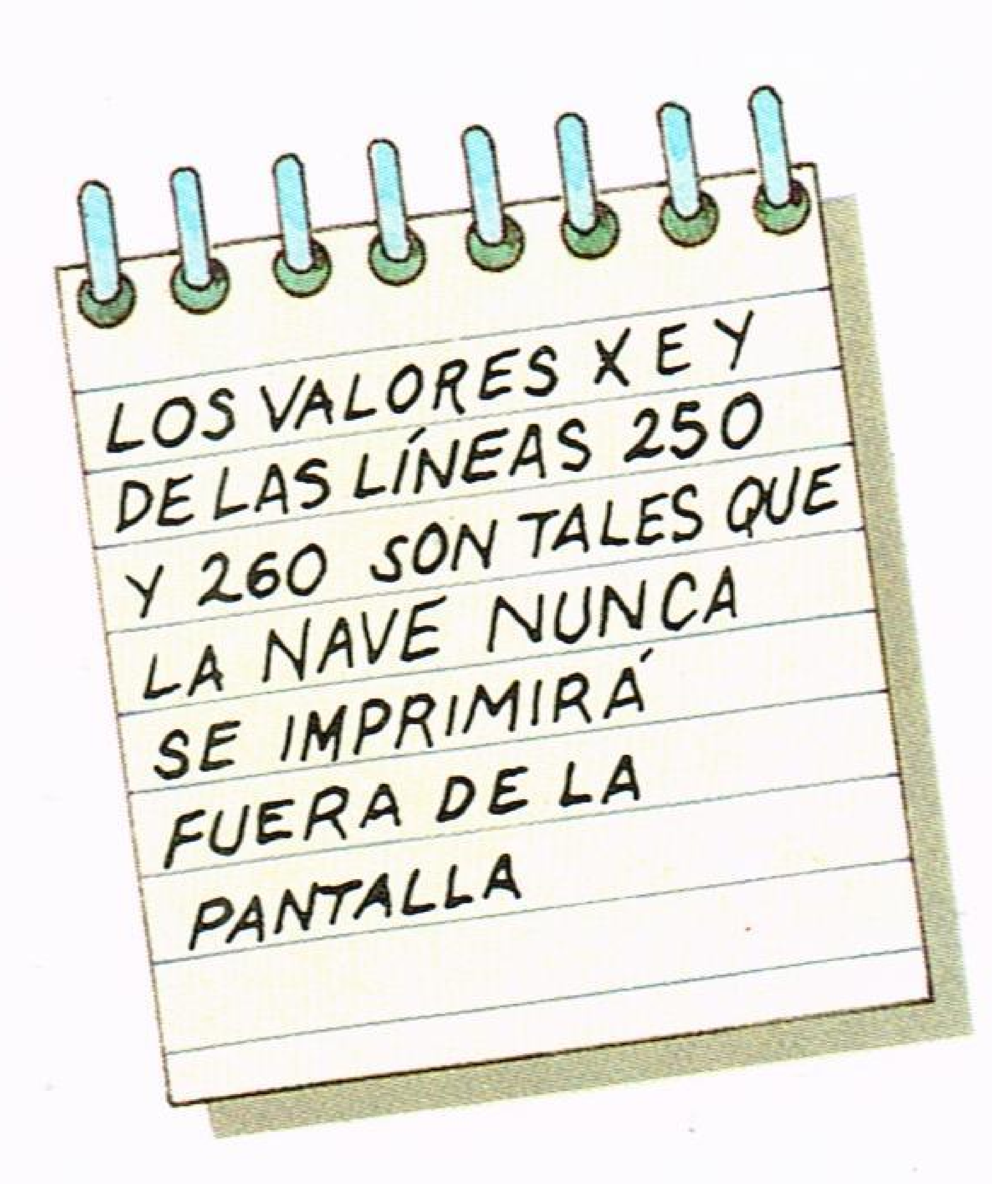


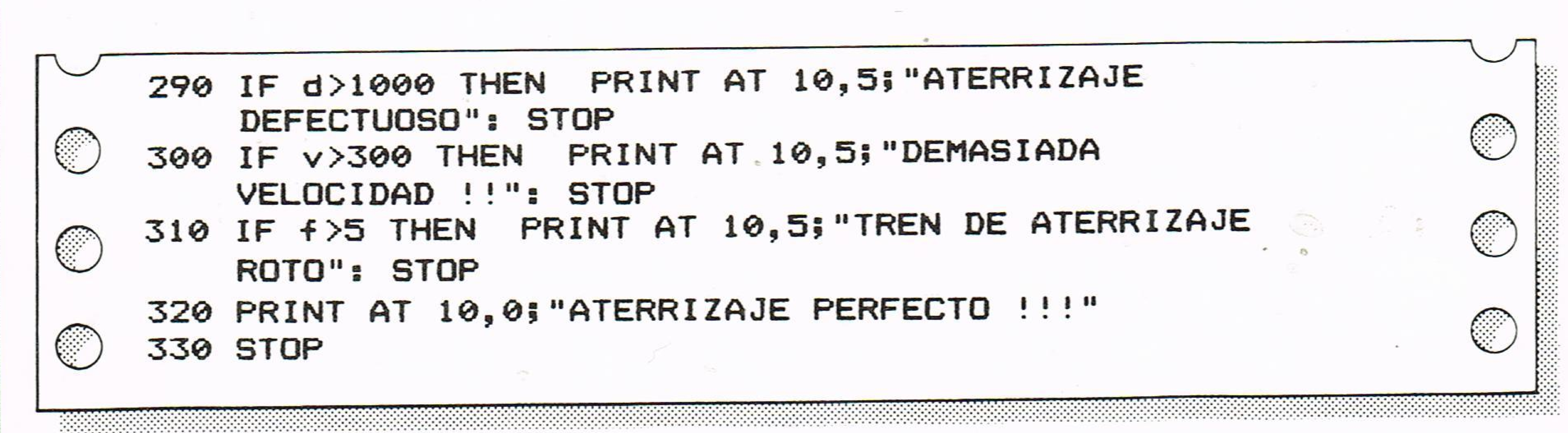
```
160 IF v<200 THEN  PRINT AT 10,5;"EL MOTOR SE TE HA
    PARADO": STOP
170 IF d<-999 THEN  PRINT AT 10,0;"EL COHETE SE HA PASADO":
    STOP
180 IF a<=0 THEN  GO TO 290
```

Las líneas 160 a 180 comprueban que la velocidad no sea demasiado baja y que la nave no se haya pasado de la pista en un valor mayor de 999. La línea 180 comprueba si la nave ha llegado a una altura cero, en cuyo caso el programa salta a la línea 290, donde una comprobación final decide si la nave ha aterrizado en la pista.

```
190 LET r$=INKEY$
200 IF r$="z" THEN  LET f=f+1: LET v=v+5
210 IF r$="m" THEN  LET f=f-1: LET v=v-5
220 IF f=0 THEN  LET f=1: LET v=v-20
230 LET a=a-f: LET v=v+(f-5)*3: LET d=d-INT (v/10)
240 PRINT AT y,x;c$
250 LET x=27-INT (d/(18000/28))
260 LET y=20-INT (a/(2000/18))
270 PRINT AT y,x; INK 1;CHR$ 142;CHR$ 140
280 GO TO 80
```

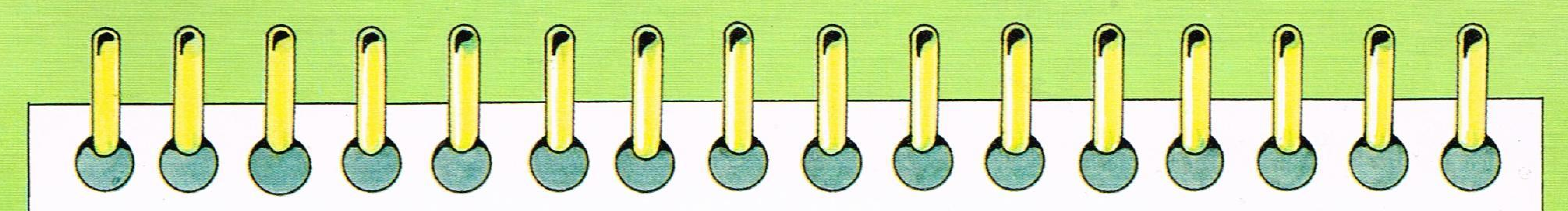
Desde la línea 190 hasta la 280, el programa permite al jugador controlar el descenso de la nave y hace los cambios en el vuelo de la nave de acuerdo con las direcciones introducidas. En la línea 190, la orden **INKEY$** le dice al ordenador que examine el teclado y vea si se ha pulsado alguna tecla. Cualquier tecla que se pulse se almacena en **r$**. Las líneas 200 y 210 comprueban si en **r$** hay una "z" o una "m" (que significan acelerar y frenar respectivamente), en cuyo caso se modifican los valores de los indicadores de acuerdo con la tecla pulsada. La orden **INKEY$** permite al programa continuar su ejecución sin tener que pararse y esperar la introducción de un dato, como hace **INPUT.** En la línea 220 el programa se asegura de que el grado de descenso nunca sea cero (si así fuera, la nave se pararía del todo) La línea 230 calcula los nuevos valores de altura, velocidad y distancia, para ser imprimidos en el panel de control. La línea 240 borra la antigua posición de la nave con los espacios almacenados en **c$.** Los nuevos valores de las coordenadas X e Y se calculan en las líneas 250 y 260, usando los nuevos valores de distancia y altura. La línea 270 dibuja la nave en su nueva posición y en la 280 el programa vuelve a buscar otra entrada.





```
290 IF d>1000 THEN  PRINT AT 10,5;"ATERRIZAJE
    DEFECTUOSO": STOP
300 IF v>300 THEN  PRINT AT 10,5;"DEMASIADA
    VELOCIDAD !!": STOP
310 IF f>5 THEN  PRINT AT 10,5;"TREN DE ATERRIZAJE
    ROTO": STOP
320 PRINT AT 10,0;"ATERRIZAJE PERFECTO !!!"
330 STOP
```

Por último vienen los mensajes de finalización. Se comprueba que el aterrizaje no sea demasiado lejos de la pista, y que ni la velocidad ni el grado de caída sean demasiado altos.



Mejora tus programas
Por último, vamos a juntar los tres programas en uno. Hay algunas modificaciones en el listado que sigue a continuación, en las líneas marcadas con un asterisco. Antes de unir los tres programas, debes cambiar los números de las líneas de los programas segundo y tercero.



Unión de los programas

En la línea 3000 se ha añadido una subrutina cuya misión es imprimir un mensaje de ALERTA en la pantalla, y algunos sonidos, cada vez que se pasa de una sección del programa a otra. En las líneas 440 y 1440 el programa llama a esta subrutina, y las líneas 430 y 1430 indican la palabra que va a aparecer en el mensaje en cada caso.

Listado de los programas

COMMODORE

```
  5 REM DESPEGUE DEL COHETE
 10 PRINT CHR$(147)
 20 PRINT SPC(13);"DESPEGUE"
 30 PRINT SPC(13);"---------QQ"
 40 PRINTSPC(3);"EL COHETE ESTA PREPARADO"
 50 PRINTSPC(3)"QQDEBES HACERLO DESPEGAR"
 60 PRINTSPC(3);"QQ■SOLO TIENES 10 INTENTOS !!■"
 70 PRINTSPC(3)"QQPOR FAVOR, TECLEA TU NOMBREQQ"
 80 PRINTSPC(3);"Y PULSA RETURNQQ"
 90 INPUTNO$
100 PRINT CHR$(147)
110 PRINT"QQBIENVENIDO A BORDO CAPITAN ";NO$
120 PRINTSPC(5);"QQPARA HACER DESPEGAR EL COHETE"
130 PRINTSPC(5);"QQDEBES ADIVINAR EL NIVEL DE "
140 PRINTSPC(5);"QQENERGIA NECESARIO"
150 PRINTSPC(5);"QQ(UN NUMERO ENTRE 1 Y 100)"
160 PRINTSPC(5);"QQTECLEA EL NUMERO Y PULSA RETURN"
170 NI=INT(RND(0)*100)
180 FORIN=1 TO 10
185 PRINT"■■QQQQQQQQQQQQQQQQQQQQ"SPC(11)" INTENTO NUMERO "IN"■QQ"
190 INPUT"     ■■■■■";NU
200 IF NU=NI THEN GOTO 280
210 IF NU <NI THEN PRINTSPC(7);NU;"..DEMASIADO BAJO, PRUEBA OTRA VEZ■"
220 IF NU>NI THEN PRINTSPC(7);NU;"..DEMASIADO ALTO, PRUEBE OTRA VEZ■"
230 NEXT IN
240 PRINTCHR$(147)
250 PRINTSPC(2);"QQQQQQQQHAS FALLADO TUS 10 INTENTOS !"
260 PRINT"QQTECLEA 'RUN'(RETURN) PARA JUGAR OTRA VEZ"
270 STOP
280 PRINTCHR$(147)
290 AL=0
```

Continuación Commodore

```
 300 OB=10000
 310 PRINTSPC(14);"QQQQELEVACION!!!!"
 320 PRINTSPC(16);"QQQQALTITUDQQQ"
 330 PRINTSPC(16);"]"AL
 340 FOR RE=1 TO 500
 350 NEXT RE
 360 AL=AL*2+1
 365 POKE 54277,15:POKE 54278,81:POKE 54296,15:POKE 54276,129
 367 POKE 54273,AL/65:FOR T=0 TO 60:NEXT
 370 IF AL<OB THEN GOTO 330
 375 POKE 54276,128
 380 PRINTCHR$(147)
 390 FORFI=1TO15
 400 PRINTSPC(11);"COHETE EN ORBITA!!!!"
 410 NEXTFI
 420 PRINT"QQ  ENHORABUENA CAPITAN ";NO$
*430 F$="ASTEROIDES"
*440 GOSUB 3000
1000 REM ASTEROIDES
1010 PRINTCHR$(147)
1020 DIM CO(50),FI(50)
1025 UD$="SQQQQQQQQQQQQQQQQQQQQQQQQQ"
1030 FORAS=1 TO 50
1040 CO(AS)=INT(RND(0)*30)+3
1050 FI(AS)=INT(RND(0)*20)+4
1055 XX=INT(RND(0)*8)+1:IF XX=6 THEN 1055
1060 POKE 646,XX:PRINTLEFT$(UD$,FI(AS))SPC(CO(AS))"[graphics]"
1065 PRINTLEFT$(UD$,FI(AS)+1)SPC(CO(AS))"[graphics]"
1070 NEXT AS
1080 TI$="000000"
1090 C$="                                        "
1100 CO=0:FI=INT(RND(0)*20)+4
1110 PRINTLEFT$(UD$,FI)SPC(CO)"X"
1120 PRINT"SQ"C$
1130 PRINTC$
1140 PRINT"STIEMPO:"MID$(TI$,3,2)" MINUTOS  ";
1145 PRINTRIGHT$(TI$,2)" SEGUNDOS"
1150 IF VAL(MID$(TI$,4,1))>2 THEN 1350
1160 INPUT "DIRECCION (NORTE,SUR,ESTE) ";DI$
1170 IF DI$="N" THEN AR=-1:DE=0:GOTO 1210
1180 IF DI$="S" THEN AR=1:DE=0:GOTO 1210
1190 IF DI$="E" THEN AR=0:DE=1:GOTO 1210
1200 GOTO 1120
1210 INPUT"DISTANCIA";DI
1220 PRINTLEFT$(UD$,FI)SPC(CO)"O"
1230 FORC=1TODI
1240 FI=FI+AR
1250 CO=CO+DE
1260 FORAS=1TO50
1270 IF CO=CO(AS) AND FI=FI(AS) THEN 1370
1280 IF CO=CO(AS) AND FI=FI(AS)+1 THEN 1370
1285 IF CO=CO(AS) + 1 AND FI=FI(AS) +1 THEN 1370
1290 NEXTAS
1300 IF FI>24 OR FI<4 THEN 1390
1310 PRINTLEFT$(UD$,FI)SPC(CO)"O"
1320 IF CO>33 THEN 1410
1330 NEXTC
1340 GOTO 1110
1350 PRINT"]QQQQQQQQQQQ       EL FUEL SE HA ACABADO"
1360 PRINT"QQ      Y QUEDARAS EN EL ESPACIO PARA SIEMPRE":STOP
1370 PRINT"]QQQQQQQQQQQ     HAS CHOCADO CON UN ASTEROIDE"
1380 PRINT"QQ    TU NAVE ESTA DESTROZADA":STOP
1390 PRINT"]QQQQQQQQQQQ      TE HAS SALIDO DE ORBITA"
1400 PRINT"QQ       Y TE PIERDES EN EL ESPACIO":STOP
1410 PRINT"]QQQQQQQQQQQ    LOS HAS ESQUIVADO BIEN!!"
```

```
 1420 PRINT"QQ    ERES UN BUEN PILOTO ESPACIAL"
*1430 F$="ATERRIZAJE"
*1440 GOSUB 3000
 2000 REM ATERRIZAJE
 2010 PRINT"♥"
 2020 D=18000:V=400:A=2000:F=5
 2030 C$="        "
 2035 UD$="SQQQQQQQQQQQQQQQQQQQQQQQQQ"
 2040 PRINT"▪   ALTURA  ◣CAIDA    ▒VELO.  ■DISTANCIA"
 2050 PRINTLEFT$(UD$,24)SPC(35)"▮R    "
 2060 X=0:Y=3
 2070 PRINTLEFT$(UD$,Y+1)SPC(X)"▥R ▀▄"
 2080 PRINT"SQQ    "C$
 2090 PRINT"▪SQQ    "A
 2100 PRINT"SQQ"SPC(13)C$
 2110 PRINT"◣SQQ"SPC(13)F
 2120 PRINT"SQQ"SPC(20)C$
 2130 PRINT"▒SQQ"SPC(20)V
 2140 PRINT"SQQ"SPC(27)C$
 2150 PRINT"■SQQ"SPC(27)D
 2160 IFV<200 THENPRINTLEFT$(UD$,11)SPC(12)"▶EL MOTOR SE HA PARADO":STOP
 2170 IF D<-999 THEN PRINTLEFT$(UD$,11)"▶ EL COHETE SE HA PASADO":STOP
 2180 IFA<=0 THEN 2290
 2190 GET R$
 2200 IF R$="A" THEN F=F+1:V=V+5
 2210 IF R$="D" THEN F=F-1:V=V-5
 2220 IF F=0 THEN F=1:V=V-20
 2230 A=A-F:V=V+(F-5)*3:D=D-INT(V/10)
 2240 PRINTLEFT$(UD$,Y+1)SPC(X)C$
 2250 X=35-INT(D/(18000/35))
 2260 Y=22-INT(A/(2000/19))
 2270 PRINTLEFT$(UD$,Y+1)SPC(X)"▥R ▀▄"
 2280 GOTO 2080
 2290 IF D>1000 THEN PRINT LEFT$(UD$,11)"▶      ATERRIZAJE DEFECTUOSO":END
 2300 IF V>300 THEN PRINTLEFT$(UD$,11)"▶      DEMASIADA VELOCIDAD!!":END
 2310 IF F>5THEN PRINTLEFT$(UD$,11)"▶      TREN DE ATERRIZAJE ROTO":END
 2320 PRINTLEFT$(UD$,11)"▶     ATERRIZAJE PERFECTO!!!":END
*3000 REM MENSAJES
*3010 FORR=1TO5000
*3020 NEXT R
*3030 POKE54276,129
*3040 PRINT"♥"
*3050 FORL=1TO20
*3060 PRINT"     ▪ALERTA! CAMPO DE "F$"!!!♥"
*3070 NEXT L
*3080 FORR=1TO8000
*3090 NEXT R
*3095 POKE54276,128
*3100 RETURN
```


SPECTRUM

```
  5 REM despegue del cohete
 10 CLS
 20 PRINT AT 4,10;"DESPEGUE"
 30 PRINT AT 5,10;"--------"
 40 PRINT AT 8,4;"EL COHETE ESTA PREPARADO"
 50 PRINT AT 10,4;"DEBES HACERLO DESPEGAR"
 60 PRINT AT 13,1; INK 2; FLASH 1;"solo tienes 10 intentos"
 70 PRINT AT 16,3;"POR FAVOR, TECLEA TU NOMBRE"
 80 PRINT AT 18,2;"Y PULSA ENTER"
 90 INPUT n$
100 CLS
```

Continuación Spectrum

```
 110 PRINT "BIENVENIDO A BORDO": PRINT "CAPITAN ";n$
 120 PRINT AT 4,1;"PARA HACER DESPEGAR EL COHETE"
 130 PRINT AT 6,1;"DEBES ADIVINAR EL NIVEL DE"
 140 PRINT AT 8,1;"ENERGIA NECESARIO"
 150 PRINT AT 10,1;"(UN NUMERO ENTRE 1 Y 100)"
 160 PRINT AT 12,1;"TECLEA EL NUMERO Y PULSA ENTER"
 170 LET nivel=INT (RND*100)+1
 180 FOR i=1 TO 10
*185 PRINT AT 16,8; INK 2; FLASH 1;"INTENTO NUMERO ";i
 190 INPUT num
 195 PRINT AT 17,16;num;"    "
 200 IF num=nivel THEN  GO TO 280
 210 IF num<nivel THEN  PRINT AT 20,1;"DEMASIADO BAJO, PRUEBA OTRA VEZ"
 220 IF num>nivel THEN  PRINT AT 20,1;"DEMASIADO ALTO, PRUEBA OTRA VEZ"
 230 NEXT i
 240 CLS
 250 PRINT AT 4,0;"HAS FALLADO TUS 10 INTENTOS"
 260 PRINT AT 6,0;"TECLEA 'RUN' (ENTER) PARA JUGAR";AT 7,0;"OTRA VEZ"
 270 STOP
 280 CLS
 290 LET altitud=0
 300 LET objetivo=10000
 310 PRINT AT 2,10;"ELEVACION !!!!!"
 320 PRINT AT 10,10;"ALTITUD"
 330 PRINT AT 10,20;altitud
 340 FOR r=1 TO 100
 350 NEXT r
 360 LET altitud=altitud*2+1
*365 BEEP .5,30
 370 IF altitud<objetivo THEN  GO TO 330
 380 CLS
 390 FOR r=5 TO 10
 400 PRINT AT r,8;"COHETE EN ORBITA !"
 410 NEXT r
 420 PRINT AT 15,1;"ENHORABUENA CAPITAN ";n$
*430 LET f$="ASTEROIDES"
*440 GO SUB 3000
1000 REM asteroides
1010 CLS
1020 DIM c(50): DIM f(50)
1030 FOR s=1 TO 50
1040 LET c(s)=INT (RND*26)+3
1050 LET f(s)=INT (RND*18)+2
*1055 LET xxx=INT (RND*7)
*1060 PRINT AT f(s),c(s); INK xxx;CHR$ 132
*1065 PRINT AT f(s)+1,c(s); INK xxx;CHR$ 135;CHR$ 130
1070 NEXT s
1100 LET col=0: LET fila=2+INT (RND*18)
1110 PRINT AT fila,col;"X"
1160 INPUT "DIRECCION (norte,sur,este) ";d$
1170 IF d$="n" THEN  LET arriba=-1: LET derecha=0: GO TO 1210
1180 IF d$="s" THEN  LET arriba=1: LET derecha=0: GO TO 1210
1190 IF d$="e" THEN  LET arriba=0: LET derecha=1: GO TO 1210
1200 GO TO 1160
1210 INPUT "DISTANCIA ";distancia
1220 PRINT AT fila,col;"O"
1230 FOR c=1 TO distancia
1240 LET fila=fila+arriba
1250 LET col=col+derecha
1260 FOR s=1 TO 50
1270 IF col=c(s) AND fila=f(s) THEN  GO TO 1370
1280 IF col=c(s) AND fila=f(s)+1 THEN  GO TO 1370
1285 IF col=c(s)+1 AND fila=f(s)+1 THEN  GO TO 1370
1290 NEXT s
1300 IF fila>20 OR fila<2 THEN  GO TO 1390
```

```
 1310 PRINT AT fila,col;"O"
 1320 IF col>30 THEN  GO TO 1410
 1330 NEXT c
 1340 GO TO 1110
 1370 CLS : PRINT AT 10,0;"HAS CHOCADO CON UN ASTEROIDE"
 1380 PRINT AT 13,0;"TU NAVE ESTA DESTROZADA": STOP
 1390 CLS : PRINT AT 10,0;"TE HAS SALIDO DE ORBITA"
 1400 PRINT AT 13,0;"Y TE PIERDES EN EL ESPACIO": STOP
 1410 CLS : PRINT AT 10,0;"LOS HAS ESQUIVADO BIEN !!"
 1420 PRINT AT 20,0;"ERES UN BUEN PILOTO ESPACIAL"
*1430 LET f$="ATERRIZAJE"
*1440 GO SUB 3000
 2000 REM aterrizaje del cohete
 2010 CLS
 2020 LET d=18000: LET v=400: LET a=2000: LET f=5
 2030 LET c$="      "
 2040 PRINT AT 1,0; INK 2;"ALTURA "; INK 4;"CAIDA "; INK 3;"VELOCIDAD ";
      INK 1;"DISTANCIA"
 2050 PRINT AT 21,28; INK 6;CHR$ 143;CHR$ 143;CHR$ 143
 2060 LET x=0: LET y=3
 2070 PRINT AT y,x; INK 1;CHR$ 142;CHR$ 140
 2080 PRINT AT 2,1;c$
 2090 PRINT AT 2,1; INK 2;a
 2100 PRINT AT 2,9;c$
 2110 PRINT AT 2,9; INK 4;f
 2120 PRINT AT 2,15;c$
 2130 PRINT AT 2,15; INK 3;v
 2140 PRINT AT 2,25;c$
 2150 PRINT AT 2,25; INK 1;d
 2160 IF v<200 THEN  PRINT AT 10,5;"EL MOTOR SE TE HA PARADO": STOP
 2170 IF d<-999 THEN  PRINT AT 10,0;"EL COHETE SE HA PASADO": STOP
 2180 IF a<=0 THEN  GO TO 2290
 2190 LET r$=INKEY$
 2200 IF r$="z" THEN  LET f=f+1: LET v=v+5
 2210 IF r$="m" THEN  LET f=f-1: LET v=v-5
 2220 IF f=0 THEN  LET f=1: LET v=v-20
 2230 LET a=a-f: LET v=v+(f-5)*3: LET d=d-INT (v/10)
 2240 PRINT AT y,x;c$
 2250 LET x=27-INT (d/(18000/28))
 2260 LET y=20-INT (a/(2000/18))
 2270 PRINT AT y,x; INK 1;CHR$ 142;CHR$ 140
 2280 GO TO 2080
 2290 IF d>1000 THEN  PRINT AT 10,5;"ATERRIZAJE DEFECTUOSO": STOP
 2300 IF v>300 THEN  PRINT AT 10,5;"DEMASIADA VELOCIDAD !!": STOP
 2310 IF f>5 THEN  PRINT AT 10,5;"TREN DE ATERRIZAJE ROTO": STOP
 2320 PRINT AT 10,0;"ATERRIZAJE PERFECTO !!!"
 2330 STOP
*3000 REM
*3010 FOR r=1 TO 1000
*3020 NEXT r
*3050 CLS
*3060 FOR l=1 TO 20
*3065 BEEP .25,40
*3070 PRINT AT 1,2; INK 2; FLASH 1;"ALERTA ! CAMPO DE ";f$
*3080 NEXT l
*3090 FOR r=1 TO 30
*3095 BEEP .25,40
*3100 NEXT r
*3110 RETURN
```


ESTA NUEVA SERIE CONSTITUYE UNA INTRODUCCIÓN A
EL ARTE DE PROGRAMAR ORDENADORES. CADA LIBRO
ENSEÑA CÓMO ESTRUCTURAR UN PROGRAMA EN
RUTINAS, Y AL MISMO TIEMPO EXPLICA Y
ANALIZA LO QUE SE LE PIDE AL ORDENADOR
QUE HAGA Y POR QUÉ.
¡Y ADEMÁS ES DIVERTIDO! CADA LIBRO CONTIENE
COMO MÍNIMO UN JUEGO DE ORDENADOR,
APASIONANTE Y ORIGINAL, ESTRUCTURADO
DE TAL MODO QUE PUEDE AMPLIARSE
O REDUCIRSE, DE ACUERDO CON EL NIVEL
O LOS DESEOS DEL USUARIO.
TÍTULOS DE LA SERIE
Gráficos - Iniciación al basic
El banco de datos